

# UNCLASSIFIED

AD NUMBER
AD487385
NEW LIMITATION CHANGE
TO Approved for public release, distribution unlimited
FROM Distribution authorized to U.S. Gov't. agencies and their contractors; Administrative/Operational Use; Jun 1966. Other requests shall be referred to Rome Air Development Center, Attn: EMLI, Griffiss AFB, NY 13440.
AUTHORITY
Rome Air Development Center ltr dtd 14 Oct 1971

THIS PAGE IS UNCLASSIFIED

487385

RADC-TR- 66-324



DISPLAY ORIENTED COMPUTER USAGE SYSTEM

TECHNICAL REPORT NO. RADC-TR- 66-324

June 1966

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMLI), GAFB, N.Y. 13440.

Rome Air Development Center  
Research and Technology Division  
Air Force Systems Command  
Griffiss Air Force Base, New York

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded, by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

## DISPLAY ORIENTED COMPUTER USAGE SYSTEM

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMLI), GAFB, N.Y. 13440.

## FOREWORD

This interim report was prepared by Informatics, Inc., Bethesda, Maryland, under contract AF30(602)-3500, Project 4594, Task 459402. The RADC program monitor was Mr. Samuel S. DiCarlo.

This work covers work performed during the period from October 1964 through April 1966. The objective was to create a capability for the on-line use of display consoles as a programming and application media.

The ultimate objective of the system is to develop tools for two types of users; a) the non-computer oriented user, usually an analyst and b) the procedure implementor, a programmer. These tools will assist both users in interacting more effectively with a computer and in using the respective capabilities of both man and machine more efficiently.

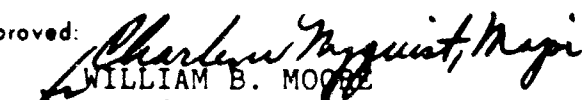
Release of subject report to the general public is prohibited by the Strategic Trade Control Program, Mutual Defense Assistance Control List (revised 6 January 1965), published by the Department of State.

This technical report has been reviewed and is approved.

Approved:

  
SAMUEL S. DICARLO  
Project Engineer

Approved:

  
WILLIAM B. MOORE  
Chief, Recon Intel Data Handling Br  
Intel and Info Processing Division

## ABSTRACT

The Display Oriented Computer Usage System (DOCUS) is a software system under which new on-line man/machine techniques for CRT console use are being designed and implemented. For the non-programming user, on-line tools such as a File Maintenance and Query Language and a Text Manipulation Language are described. For the programming user a capability for on-line programming and debugging through the use of FORTRAN, CODAP, and a Display Oriented Language (DOL) is provided by the system. The basic framework in which these user tools operate is also provided by two groups of generalized function key programs. One group of keys, the General Operating Language gives the non-programming user a display manipulation, library control, and system communication facility. The second group of keys, the Procedure Implementation Language, allows the programmer to create displays and key programs and to build a repertoire of functions in a system and various user libraries.

## TABLE OF CONTENTS

1.0	INTRODUCTION	1- 1
1.1	BACKGROUND	1- 2
1.2	OPERATING PRINCIPLES	1- 2
1.3	USER ASPECTS	1- 7
2.0	DOCUS CONCEPT	2- 1
2.1	OPERATING SYSTEM	2- 3
2.1.1	Introduction	2- 3
2.1.2	CO-OP Monitor	2- 5
2.1.2.1	CO-OP Software Hierarchy	2- 5
2.2.2.2	Integration of DOCUS into CO-OP	2- 6
2.1.3	DOCUS Executive	2- 6
3.0	LANGUAGES	3- 1
3.1	EXECUTIVE LANGUAGES	3- 1
3.1.1	Procedure Implementation Language (PIL)	3- 1
3.1.1.1	PIL Overlay Orgainzation	3- 2
3.1.1.2	General Control	3- 2
3.1.1.3	Data Manipulation	3- 4
3.1.1.4	Library Maintenance	3- 5

3.1.2	General Operating Language (GOL)	3- 8
3.1.2.1	GOL Overlay Organization	3- 8
3.1.2.2	General Control	3- 9
3.1.2.3	General Communication	3- 9
3.1.2.4	Display Manipulation	3-10
3.1.2.5	Library Control	3-11
3.1.3	Language Editing Overlay (LEO)	3-13
3.1.3.1	Editing	3-14
3.1.3.2	Code Scanning	3-16
3.1.3.3	Access and Storage of Programs	3-17
3.2	PROGRAMMING LANGUAGES	3-19
3.2.1	Display Oriented Language (DOL)	3-19
3.2.1.1	Introduction	3-19
3.2.1.2	Objectives	3-20
3.2.1.3	Generalization of Functions in Man-Machine Communi- cation	3-21
3.2.1.4	Generalization of a Display Console	3-22
3.2.1.4.1	Display Console Features	3-22
3.2.1.4.2	The BR85 Console Measured Against this Generic Con- sole	3-23



3.2.1.5	Proposed Realization of DOL Capability	3-24
3.2.1.6	DOL Statement Formats	3-26
3.2.1.7	DOL Statement Classes	3-26
3.2.1.8	Typical DOL Statements	3-27
3.2.2	FORTTRAN Language Overlay	3-31
3.2.2.1	Introduction	3-31
3.2.2.2	Coding Aids	3-32
3.2.2.3	Keys	3-33
3.2.3	CODAP	3-39
3.2.3.1	Introduction	3-39
3.2.3.2	Instruction Tutorial Keys	3-40
3.2.3.3	Subroutine I/O Keys	3-43
3.2.3.4	Control Keys	3-43
3.3	APPLICATION LANGUAGES	3-47
3.3.1	File Management	3-47
3.3.1.1	Introduction	3-47
3.3.1.2	File Definition Phase	3-48
3.3.1.3	File Maintenance Phase	3-51
3.3.1.4	Query Phase	3-54
3.3.2	Text Manipulation	3-56
3.3.2.1	Introduction	3-56
3.3.2.2	Current Capabilities	3-57
3.3.2.3	Further Extensions of Overlay	3-65

## **APPENDICES**

**APPENDIX A - Definitions**

**A-1**

**APPENDIX B - DOCUS Hardware Configuration**

**B-1**

**APPENDIX C - Operation Under DOCUS Executive**

**C-1**

**APPENDIX D - References**

**D-1**

## 1.0 INTRODUCTION

The development of techniques for on-line information processing and man-machine communication has become an important area of the computing world. As a part of the Display Oriented Computer Usage System (DOCUS) effort, techniques are being developed to place more data processing capability within the reach of computer users whose areas of interest may be related to information retrieval, scientific research, command and control, business management, or any area where algorithms are not well developed. It is desirable to make these on-line users less dependent upon the professional programmer, and to reduce the implementation time for programming new applications.

The Display Oriented Computer Usage System is a fundamental software system which facilitates:

- A) Implementation of new man/machine operations and procedures;
- B) On-line development of sophisticated processing functions from simpler functions;
- C) On-line programming and debugging through the use of higher order languages.

The basic objective of DOCUS is to provide a software system which allows the development of methods for the solution of day-to-day problems in an on-line environment. In typical operational situations, the approach to problem solving and the development of new procedures must be handled in an ad hoc manner as the situation develops. The ability of

DOCUS to work in this environment is an important characteristic. There are other on-line systems which are basically on-line programming systems and are not directed to the non-computer oriented user. DOCUS has the on-line programming capability plus the techniques and methods to help the analyst/user and the procedure implementor/user solve his individual problem.

## 1.1 BACKGROUND

There have been several efforts, which attempted to develop or are currently developing, man-machine communication techniques. Recognizing the impressive list of time-sharing systems in existence and their remote terminals, the references under consideration here are restricted to systems using display consoles in a general purpose, conversational or interactive mode with the human operator. In 1960 the system design for DODDAC (1) used display consoles in a conversational mode. An on-line computer center (2) by Culler-Fried and the related TRW two-station on-line scientific computer (3) used the man-machine concept for scientific problem solving. Work for the National Military Command System Support Center (4), SDC's general purpose display system (5), and AESOP (6) are contributing to man-machine techniques in information retrieval. SKETCHPAD (7), General Motors DAC-1 System (8), and IBM's Alpine System (9) made contributions in graphical design. It is upon this base and a recognition of these systems that the current DOCUS work was commenced.

## 1.2 OPERATING PRINCIPLES

An important principle that is embodied in DOCUS is that the user, through the use of a visual display device, may communicate with a computer system if he has available to him the following:

- A) Appropriate languages.
- B) Appropriate equipment.
- C) An operating system.

Current available computers and visual display devices are quite adequate for the user's needs. What has been lacking are the languages and the operating system which tie various languages together, interface with the equipment, and provide a reference for the user's operations.

In the DOCUS concept this man-machine communication is accomplished by a series of related steps. First, the user performs an action (pressing a function key, uses a light pen, or enters data via a keyboard) (Appendix A provides a list of basic definitions of frequently used terms in this report); the computer responds to the user's actions and projects a display or turns on or off some indicator lights; the user examines the computer's messages and performs another action. The process continues until the user is satisfied, an error occurs, or the computer is unable to respond to the user in a satisfactory manner. See Figure 1-1.

The individual action which the user initiates is defined as a function. A group of related functions are combined to make a procedure. A group of related procedures are similarly combined to make a language. Typically, a function would consist of a discrete sequence of computer instructions which are initiated by a console function key and a key group would correspond to an "overlay" on a display console.

DOCUS includes capabilities for easily examining alternate problem approaches, on-line programming and debugging, on-line program documentation and user guidance and dynamic user interaction with

USER

COMPUTER / CONSOLE

PRESS FUNCTION KEY

IDENTIFIES FUNCTIONS  
AND PRESENTS DISPLAY

EXAMINES DISPLAY,  
ENTERS DATA OR SELECTS  
ITEM WITH LIGHT PEN

PROCESSES DATA AND  
NOTIFIES ANALYST  
OF RESULTS WITH A  
DISPLAY OR STATUS LIGHT

OBSERVES RESULTS, CON-  
TINUES ANOTHER ACTION

Man/Machine Interaction

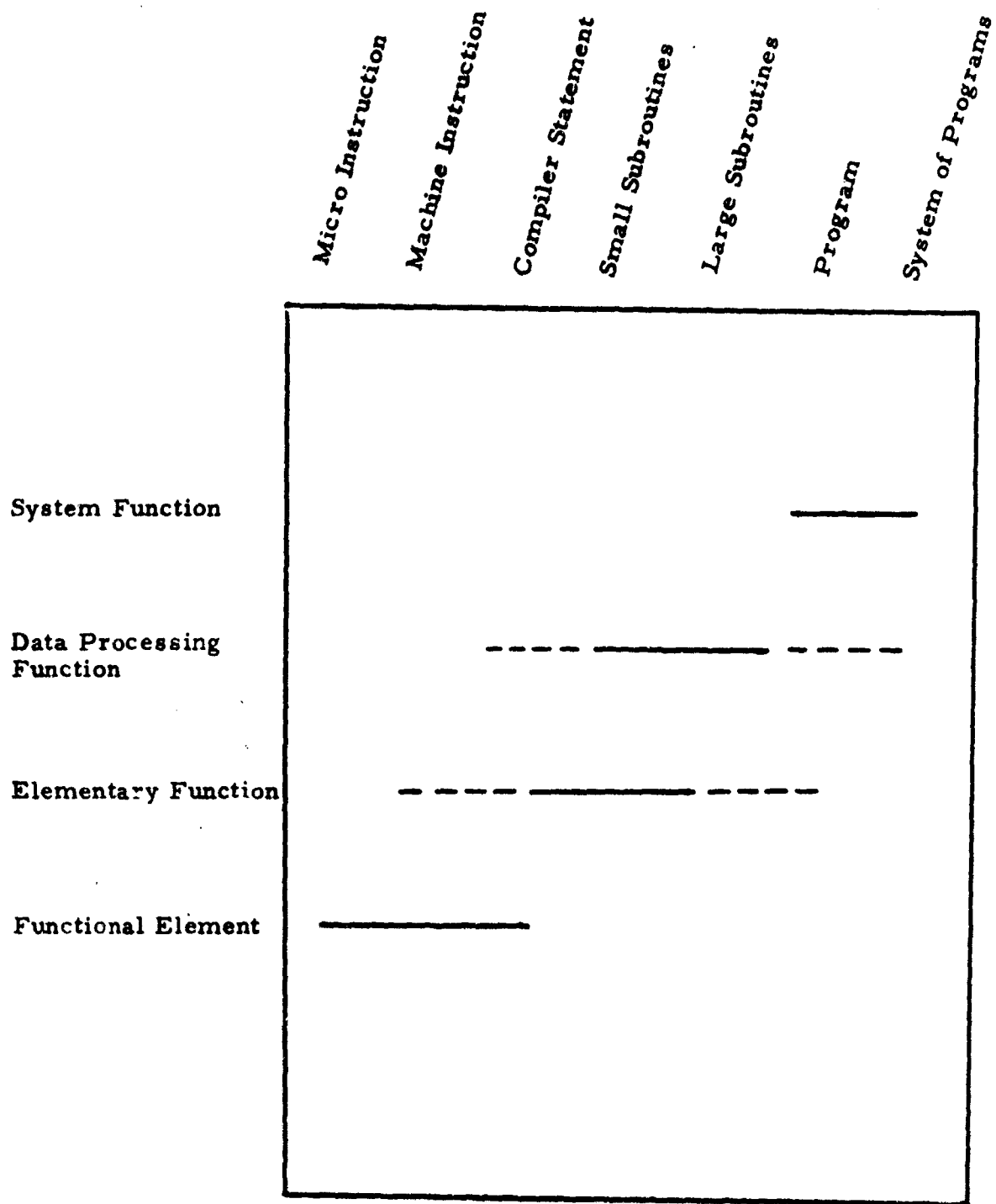
Figure 1-1

executable programs. To understand how this may be accomplished, it is first necessary to consider some basic elements in a problem solution.

When solving different problems, there are often elements common to these problems. One example is the use of an I/O subroutine. These common elements may be basic logical operations such as seek a display, add an item to a list, or read a record. They will be referred to as elementary functions. The level of complexity of these functions can best be shown by Figure 1-2. These functions are usually more than one machine instruction and hardly ever as large as a program. Usually these functions are several instructions or a small subroutine. The solid line presents the normal range of complexity of these functions and the dashed lines represent the possible limits of complexity.

In the design phase of an approach, individual elementary functions are usually identified. If enough common functions can be identified then the solution to a new problem can be simplified by using elementary functions developed during the course of solving previous problems. If these elementary functions can be combined and executed in a man-machine environment then there should be an improvement in the time required to solve a problem because of reduced implementation time. A second benefit may accrue to the user of an on-line system. Because the user interacts very closely with the solution as it is developed, unexpected events may be detected and ad hoc decisions made which redirect the approach to the solution.

With a well designed system, and a sufficiently large number of flexible, elementary functions available in a library, the control in solving a problem will no longer be with the programmer but with the non-programming user. He will now be "programming" (for want of a better word) the system, but he will be doing it in his own application oriented language and not a machine or compiler language. Toward these ends



Complexity of Functions

Figure 1-2



the basic objectives of DOCUS may be summarized as follows:

- A) Design new techniques of man/machine communications;
- B) Improve existing man/machine techniques;
- C) Develop display console utilization methods;
- D) Implement and test selected man/machine techniques.

### 1.3 USER ASPECTS

DOCUS recognizes two classes of users: the analyst, and the procedure implementor. The analyst is the ultimate customer for whom the hardware/software system exists; he is the problem solver and he is the person who interacts with data. He is least likely to know or care much about details of data processing hardware or software since he looks upon the computer as a tool. Furthermore, he prefers an application oriented language rather than any stylized computer language.

The procedure implementor is a data processing technician or programmer who understands the analyst's applications sufficiently that he can adequately design and implement elementary functions for use by the analyst. As such he will also understand computer programming, the processor hardware, and the techniques of man-machine communication.

DOCUS provides two categories of techniques to meet the system objectives. These techniques enable the analyst and the procedure implementor to operate more efficiently. First, the procedure implementor has at his disposal a variety of man-machine communication

techniques and a Display Oriented Language (DOL). The Display Oriented Language is used to express application procedures and implement them for the analyst within an operating system. These procedures may be complete programs such as a query and edit or a mathematical analysis; or they may consist of data processing functions such as sorts, report generation, and data conversion. The procedures may also consist of more elementary functions which are application oriented such as combine lists, remove item from list, order list alphabetically, identify a record, or show a display. The Display Oriented Language consists of commands which govern the step-by-step process of man-machine communication together with necessary operations for creating and manipulating operational elements, which in this case are displays and functions. In addition, the procedure implementor has available for use, in an on-line mode, debugging and utility programs.

Second, the analyst, having at his disposal basic building blocks of procedures and functions provided by the procedure implementor, may manipulate these operations on line by combining, modifying, and relating them so as to generate more complicated functions and procedures for his use. In this manner he employs terminology and operations familiar to him. Hence, the analyst can generate his own personalized processing capability.

Both of these capabilities are essential to achieving more efficient use of data processing technology. The dichotomy of users is considered in DOCUS since it facilitates discussion and organization of the material. It is realized that many users are in both categories and nothing in DOCUS precludes the use of the system by the user in both categories.

## 2.0 DOCUS CONCEPT

The concept of problem solving, scientific, data processing, or any other classification, through DOCUS involves the use of elementary functions, displays, and human decisions. This is best characterized as a man stimuli-machine response - machine stimuli-man response .... system. This sequential transfer of action control between the user and the computer requires that the user's application oriented functions and the system's executive functions be essentially small steps in the logical problem solving chain.

In a general man-machine situation, the basic system is developed around a multiple display console configuration, thus allowing rapid communication between users as well as between the users and the processor. Such a configuration requires, for efficient operation, suitable hardware including on-line terminals and mass storage devices. The CRT console as a terminal, offers the advantage of speed, quiet operation, and additional features, such as line drawing and light pen, over the conventional teletype terminals.

To properly control a complex hardware system (Appendix B) of this type requires an efficient operating system. This is described in section 2.2. An understanding of DOCUS requires knowledge of the library and system oriented overlays for the display consoles. These are described in later sections. A summary description of the Display Oriented Language is also given. Figure 2-1 shows the relation between the various elements of DOCUS and the users.

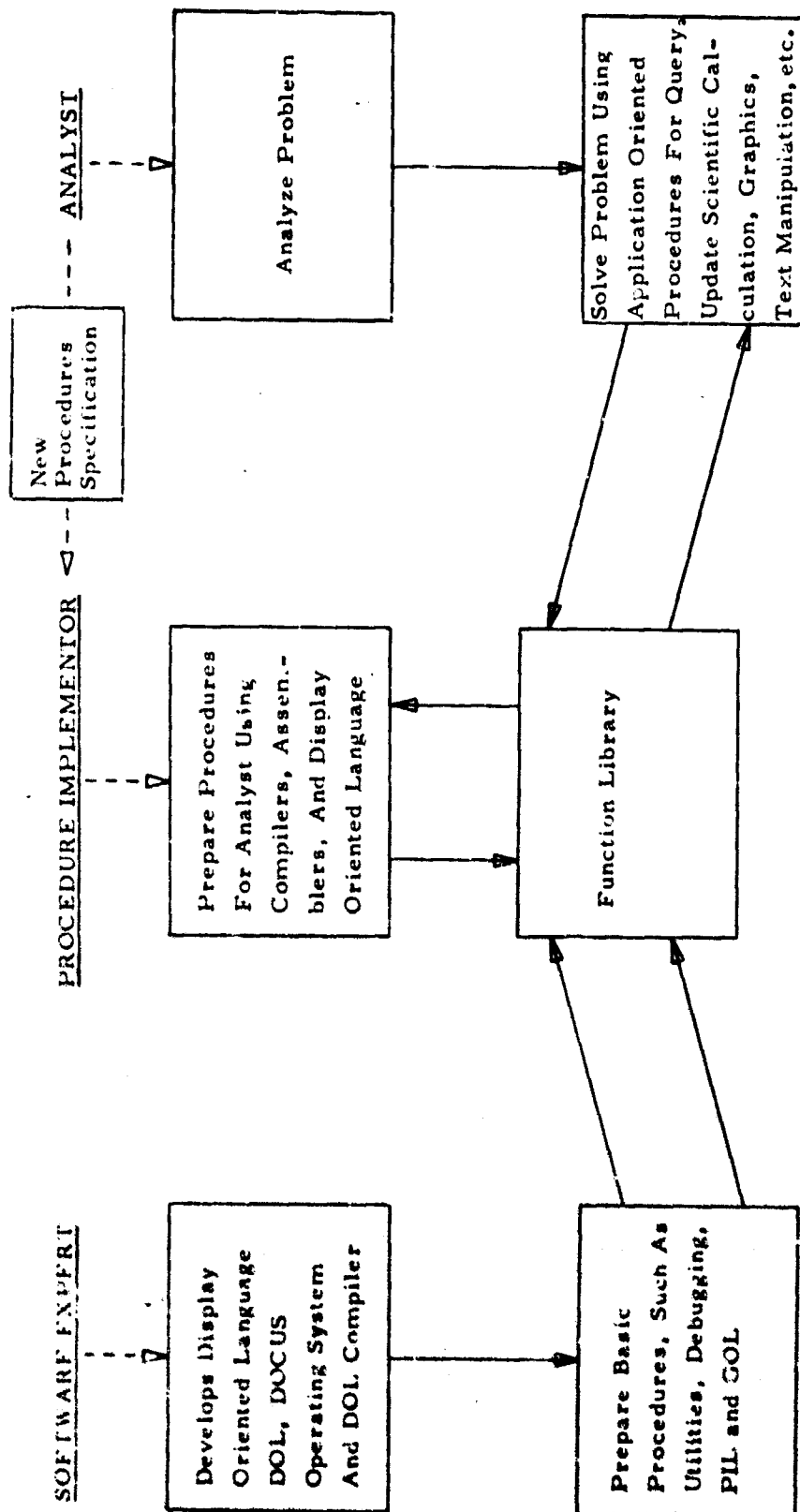


Figure 2-1. DOCUS Elements

## 2.1 OPERATING SYSTEM

### 2.1.1 Introduction

In DOCUS, the operator is allowed to call at will any one of a number of programs called overlays. These overlays, in turn, are composed of programs called key programs. These key programs are organized into branching logical sequences such that after the execution of any key program, there will be a set of key programs that it "makes sense" to execute next; the operator is allowed to execute any program in this set by pressing the corresponding key. The key programs themselves may call on other programs called subroutines as well as on each other. Moreover, key programs may communicate data to other key programs via common areas; communication is allowed not only between the key programs of an overlay but also between the key programs of two different overlays provided they are intended to work together. Finally, it should be noted that DOCUS allows the user to define new key programs and overlays and to replace an existing version of a key program by a modified version; the user may also alter the position of a key program in a logical sequence; all of these additions may be done on-line under control of the DOCUS system.

Many key programs will find it necessary to communicate with the BR85 display console. They will need to 1) transmit data to the BR85 for display on the screen, 2) read data from the BR85, 3) turn on (off) BR85 lights, and 4) determine the location of a character that has just been light-penned by the operator, etc.

From the preceding discussion it is evident that there are a certain number of general control and utility functions that are required for the functioning of most key programs and overlays. Where possible, these functions have themselves been incorporated into key programs or

into subroutines which are available to all key programs but are only loaded into the computer when needed by an overlay that is called by the operator. However, there are certain functions that are used so widely, e. g. , interpretation of BR85 interrupts, communication with the BR85, loading of overlays, calling on a key program, idling when there are no jobs, magnetic tape, I/O, that it is found convenient to incorporate them into a part of DOCUS called the Operating System that is always loaded into the computer at the beginning of DOCUS operation. Furthermore, certain functions are employed so frequently that it is desirable to provide simple shortcuts to their use. For example, any key program can turn on a set of key lights. However, a standard set, called the initial lights is defined for each overlay; if a key program exits to the operating system in a simple standard way, this set is automatically turned on by the operating system itself. The operating system simplifies the task of writing key programs, without in any way limiting the capability of the key program writer.

Thus the operating system provides a number of basic services:

- A) It initializes the system hardware and the DOCUS system so that the user can begin to use DOCUS;
- B) It provides a number of functions used by most key programs. This includes the basic function of responding to a key depression by calling on the corresponding key program as well as the various computer-display console communication subroutines used by the key program;
- C) It provides conventional shortcuts for certain often-used functions.

- D) It provides a standard "control center" to which every key program returns when it is done or when it is waiting for some operator action.

The operating system serves both types of users. For the analyst it provides continuity between his actions and the basic routines concerned with the retrieval of displays and the execution of functions. For the implementor, the operating system is the framework under which the implementation cycle for new procedures can be reduced. Incorporated in the system are all of the bookkeeping, transfers, display manipulation, input/output control normally of concern to the procedure implementor.

The operating system provides the overall control to handle interrupts from the console and perform general display manipulation operations in addition to library processing.

#### 2.1.2 CO-OP Monitor

Since the principal efforts of DOCUS are directed to man/machine communications, it was decided to minimize the effort expended on general system utility functions. This has been accomplished by using the existing CO-OP Monitor system. DOCUS has at its disposal the full range of software features provided by the CO-OP Monitor. These include the FORTRAN compiler, the CODAP assembler, the interrupt controller, the Input-Output controller, standard I/O drivers, and the relocatable binary loader.

##### 2.1.2.1 CO-OP Software Hierarchy

The standard 1604 system is composed of the following hierarchy of software:

- A) Master Control System (MCS) - Composed of loader, job sequencer, and I/O controller;
- B) Subordinate Control Systems (SCS) - CO-OP is a system in this category;
- C) Compilers and CODAP.

#### 2.1.2.2 Integration of DOCUS into CO-OP

DOCUS is integrated with the above system in a simple manner. Logically, the DOCUS Executive is treated as an SCS; the name DOCUS simply replaces the name CO-OP in the appropriate field of the MCS control card. Physically, the MCS and DOCUS are assembled and loaded as one program. Thus, the DOCUS Executive is loaded into computer memory at the same time as MCS; however, control goes first to MCS which then passes control to DOCUS when it encounters the name DOCUS on the MCS control card.

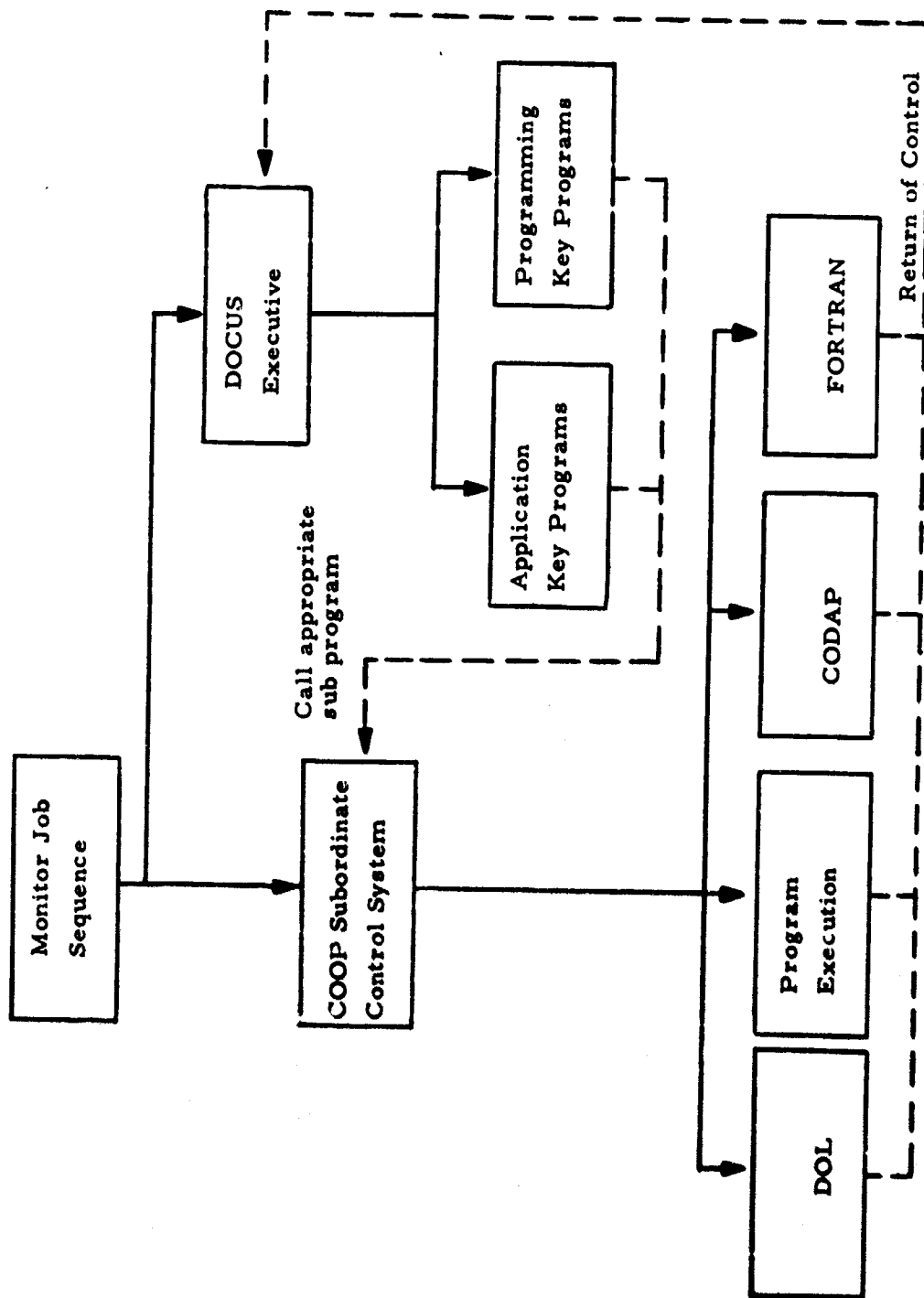
The DOCUS Executive differs from an ordinary SCS like CO-OP in one important way. CO-OP is a job processor which returns to MCS when it has no more jobs to process. DOCUS, on the other hand, retains control indefinitely; if there are no console requests for action, it idles in its own idle loop rather than returning to the MCS idle loop.

If the MCS card specifies a non-DOCUS job, the internal core used by the DOCUS Executive is released and subsequent operations proceed in the standard mode (Figure 2-2).

#### 2.1.3 DOCUS Executive

The DOCUS Executive is the link between display console and 1604 computer. It provides the means by which control is passed





DOCUS - COOP  
Figure 2-2

from one console to another (in a multi-console environment), from one overlay to another, and from one key program to another.

The Executive accepts, interprets, and acts upon interrupts from the BR85 display console. In response to a given interrupt message, it may do one of the following: 1) record an overlay number, 2) load the key programs and subroutines associated with a given overlay number, 3) execute a specified key program, 4) record the location in BR85 memory of a character that has just been light-gunned and the value of the character itself.

The Executive acts as a central control for all key programs. When a key program completes its function, it always exits to the Executive. If the key program is not complete but is waiting for some operator action to be completed, it returns to the Executive; when the operator signals that his action is complete, the Executive returns control to the waiting key program.

A more detailed description of the operation of the Executive, basic subroutines, and communication areas will be found in Appendix B of this report.

### **3.0 LANGUAGES**

The basic element of any DOCUS language is the key program which is designed to perform a given function. The set of keys which together enable the programmer/analyst to build and solve a series of related problems are put together into an overlay which can be called upon as desired by the DOCUS Executive. The overlay standing alone defines a DOCUS language.

In DOCUS three types of languages have been implemented:

- A) Executive
- B) Programming
- C) Applications

#### **3.1 EXECUTIVE LANGUAGES**

An executive language is defined as a language that provides a method of handling functions related to general system operation. Two specific executive type languages have been developed for the DOCUS system. These languages are the Procedure Implementation Language (PIL), which is programmer oriented, and the General Operating Language (GOL), which is analyst directed. While an overlap of common requirements does exist between the analyst and programmer, there are sufficient additional requirements, as in the methods of controlling the system, to justify the implementation of an executive language for each major type of user.

### **3.1.1      Procedure Implementation Language (PIL)**

The Procedure Implementation Language (PIL) provides the user, programmer or procedure implementor, with a vehicle for controlling and directing various general system functions. An analogy can be drawn between many of the function keys provided and the control cards used in a batch-processing oriented system.

Through the use of function keys the PIL user is given direct control over the system in the areas of light control, display manipulation, program manipulation, and library generation and modification.

#### **3.1.1.1      PIL Overlay Organization**

The functions of the PIL overlay have been divided into three sections. These sections are General Control, Data Manipulation, and Library Maintenance. Figure 3-1 illustrates the organization of the PIL overlay function keys.

#### **3.1.1.2      General Control**

The General Control keys provide the user with the ability to manipulate console lights, retrieve displays, construct displays, and transmit messages.

- A) START - used to activate the overlay and accomplish the initialization of communication words used by key programs on the PIL overlay. On completion of the initialization process the function key lights are set to the on condition and a display is projected on the CRT informing the user that the overlay is ready for use.**

- B) **TURN ON LIGHT** - used to turn on any function key or status light. This key presents the user with a format display into which he must insert the appropriate information.
- C) **TURN OFF LIGHT** - reverse light logic of **TURN ON LIGHT**.
- D) **DISPLAY TO WORK TAPE (DTW)** - transfers a display from the DOCUS library tape to the DOCUS work tape. It may also simultaneously delete the display from the library and project the first page on the CRT. Any series of coding form displays are placed under control of the editing overlay for processing.
- E) **SEND MESSAGE** - allows the user to enter a message via the console keyboard. The message is converted and transferred to the computer console typewriter.
- F) **SEEK AND DISPLAY** - allows the user to recall a display, by display name from the DOCUS library. The first page of the display is automatically projected on the CRT.
- G) **BEGIN DISPLAY** - rewinds the DOCUS work tape and initializes page control items in preparation for the generation of a new display.
- H) **SAVE PAGE** - transfers a displayed CRT page to the DOCUS work tape. The saved pages of a display may be added to the library via the **ADD DISPLAY TO LIBRARY** key or altered via the **CORRECT PAGE** key.

#### **3.1.1.3 Data Manipulation**

These keys perform such tasks as generating list displays, page manipulation, and assisting in the processing of various assembly operations.

- A) LIST KEY NAMES** - provides user with the ability to request a listing on the CRT of key program names and key assignments for any given overlay.
- B) LIST DISPLAYS** - provides a listing on the CRT of all current displays residing in the DOCUS library.
- C) RECEIVE MESSAGE** - allows the user to display the contents of the message buffer on the CRT. Any key program, user program, subroutine, or the Executive may place data in the buffer.
- D) ASSEMBLY OUTPUT CONTROL** - used in conjunction with the FORTRAN and CODAP overlays under DOCUS to save the original coding pages and assembled listings of processed programs. They are saved as named displays.
- E) LIST PRINT TAPE** - used for generating and projecting a display created from any information written on the standard system output tape, e.g., system comments, listings, user program output.

- F) **NEXT PAGE** - used to manipulate multi-page displays. Projects the page following the one being displayed currently.
- G) **PREVIOUS PAGE** - similar to **NEXT PAGE** except it projects the page preceding the current display.
- H) **SCAN FORWARD** - produces the serial projection of the pages of a multi-page display, holding each page for about 3 seconds.
- I) **SCAN BACKWARD** - operates the same as **SCAN FORWARD** except the motion is reversed.
- J) **CORRECT PAGE** - allows the operator to rewrite display pages. The last page of data projected is replaced by the contents of the console memory. **CORRECT PAGE** may not be used to update the **DOCUS** library.

#### 3.1.1.4 Library Maintenance

The Library Maintenance keys provide a method of creating or editing a **DOCUS** library of key programs and displays.

Most of the function keys in this category present format displays into which the user must enter information such as program names and initial light conditions and overlay numbers and keys.

- A) **ADD PROGRAM TO LIBRARY** - used to either add a program to the existing library tape or to generate a new library tape which includes the new program. In either case the new program must be available on the **CO-OP Load & Go** tape (automatic under **DOCUS**).

- B) **ADD DISPLAY TO LIBRARY** - used to transfer displays from the DOCUS work tape to the DOCUS library tape.
- C) **ASSIGN PROGRAM KEY** - permits assignment of named key programs to overlays other than the ones for which they were originally implemented. This eliminates the need for multiple copies of the same key program.
- D) **DELETE LIBRARY** - allows the user to delete items in the DOCUS library by nullifying the appropriate library directory entry. This prevents the use of unwanted or out-of-date items.
- E) **EDIT DOCUS LIBRARY** - generates a copy of the DOCUS library removing any items previously indicated as deleted by the **DELETE LIBRARY** key program. The copy will be on the DOCUS work tape.
- F) **DUPE DISPLAY (RENAME)** - provides the user with a tutorial display that specifies the exact steps to take in order to copy and rename any display that is currently on the DOCUS library tape.



# RIGHT PROGRAM KEY OVERLAY

OVERLAY NR 03

OVERLAY NAME P.I.L.

PREPARED BY

START	P-31		P-32	TURN ON LIGHT	P-33	TURN OFF LIGHT	P-34	DISPLAY TO WORK TAPE	P-35
LIST KEY NAMES	P-36	LIST DISPLAY NAMES	P-37		P-38	ENTER	P-39	ASSIGN PROG. TO KEY (N)	P-40
EDIT DOCUS LIBRARY	P-41	DUPE DISPLAY (RENAME)	P-42		P-43	ADD DISPLAY TO LIBRARY	P-44	ADD PROGRAM TO LIBRARY	P-45
SEEK DISPLAY	P-46	DELETE FROM LIBRARY	P-47	SEND MESSAGE	P-48	RECEIVE MESSAGE	P-49	ASSEMBLY OUTPUT CONTROL	P-50
NEXT PAGE	P-51	PREVIOUS PAGE	P-52	SCAN FORWARD	P-53	SCAN BACKWARD	P-54	CORRECT PAGE	P-55
	P-56	BEGIN DISPLAY	P-57	SAVE PAGE	P-58	END DISPLAY	P-59	LIST PRINT TAPE	P-60

Figure 3-1

### 3.1.2 General Operating Language (GOL)

The General Operating Language (GOL) provides the basic framework in which a user may solve problems through the application of previously defined functions. The GOL overlay contains both general command and system functions that most users require to interact with any of the application language overlays.

A close resemblance exists between the functions of the PIL and GOL overlays. Special note should be taken that the PIL overlay is programmer oriented while the GOL overlay is analyst oriented. Both types of users, programmer and analyst, have a need to perform many similar tasks. For example, both have a need to manipulate displays, send and receive messages, and interact with a library of displays and key programs. However, there is a difference in the level of usage that can be expected considering the two classes of users. It is reasonable to assume that an analyst may generate and save displays and not have a need to manipulate key programs. On the other hand, a programmer normally requires both of these capabilities in achieving a problem solution.

#### 3.1.2.1 GOL Overlay Organization

The function keys on the GOL overlay have been classified into four sections. These sections are defined as being:

- A) General Control
- B) General Communication
- C) Display Manipulation
- D) Library Manipulation

In the discussion of these areas, those function keys identified by \* have been previously discussed in the sections of PIL. Figure 3-2 illustrates the organization of the GOL overlay.

#### 3.1.2.2 General Control

The function keys that are included in this section are as follows:

- A) START - activated to initiate loading of the overlay.
- B) ENTER - pressed to signal completion of a requested command (usually requested via a format display output from a key program).
- C) SELECT - allows the user to select up to 30 items in a list display by light-gunning the asterisk which precedes each entry in a list display. The "\*" is changed to an "X" to indicate selection. A table of information about the selected items is constructed.
- D) REJECT - allows the user to delete a selection made with the SELECT key. This is accomplished by light-gunning the "X" preceding the item to be deleted.

#### 3.1.2.3 General Communication

The General Communication function keys permit the user to send messages to the computer console typewriter and receive messages from a standard system buffer. Both key programs and the computer operator have the capability of placing messages in the standard message buffer.

- A) SEND MESSAGE \*
- B) RECEIVE MESSAGE \*

#### 3.1.2.4 Display Manipulation

The Display Manipulation function keys provide a means of constructing and saving displays. The function keys listed below enhance display operations:

- A) BEGIN DISPLAY \*
- B) SAVE PAGE \*
- C) END DISPLAY \*
- D) NEXT PAGE \*
- E) PREVIOUS PAGE \*
- F) SCAN FORWARD \*
- G) SCAN BACKWARD \*
- H) CORRECT PAGE \*
- I) SEEK DISPLAY \*
- J) SAVE DISPLAY - used to retain temporary information, freeing the CRT for some alternate function.
- K) RESTORE DISPLAY - used to retrieve a page that was temporarily stored via the SAVE DISPLAY key.

### 3.1.2.5 Library Control

The keys in this category are used to delete displays, move displays, and request various list displays.

- A) LIST DISPLAYS \*
- B) LIST KEY NAMES \*
- C) DELETE LIBRARY \*
- D) DISPLAY TO WORK TAPE \*

# RIGHT PROGRAM KEY OVERLAY

OVERLAY NR 01

OVERLAY NAME G.O. L.

PREPARED BY \_\_\_\_\_

START	P-31		P-32		P-33		P-34	DISPLAY TO WORK TAPE	P-35
SELECT	P-36	REJECT	P-37		P-38	ENTER	P-39	ADD DISPLAY TO LIBRARY	P-40
SEND MESSAGE	P-41	READ MESSAGE	P-42	LIST DISPLAYS	P-43	LIST KEY NAMES	P-44		P-45
DELETE LIBRARY	P-46	SEEK AND DISPLAY	P-47	SAVE DISPLAY	P-48	RESTORE DISPLAY	P-49		P-50
PREVIOUS PAGE	P-51	NEXT PAGE	P-52	SCAN FORWARD	P-53	SCAN BACKWARD	P-54	CORRECT PAGE	P-55
BEGIN DISPLAY	P-56	SAVE PAGE	P-57	END DISPLAY	P-58		P-59	LIST PRINT TAPE	P-60

Figure 3-2

### 3.1.3 Language Editing Overlay (LEO)

It has been stated previously that a major thesis in the DOCUS concept is that of man/machine interplay, and one can easily visualize this interplay in the PIL and GOL overlays. However, whereas these overlays serve as external structures assisting the analyst/procedure implementor, it soon becomes apparent that internal assistance is also needed.

The basic medium for problem solving continues to be that element designated as a key program, written in an assembler or compiler type language (DOL, CODAP, FORTRAN). Few procedure implementors consistently describe a procedure in error free code. Error free code is defined as a set of statements which when successfully translated into machine instructions performs the function intended without further modification. The process of creating this error free code usually involves several sets of modifications to the original code and subsequent recompilations. Also, a system may be on a daily production run basis, but it may be desirable to change the system to provide additional capabilities at some time long after its inception. The above functions must be accomplished without completely rewriting the procedure each time, or the capability becomes useless.

A need therefore arises for a system which provides for the following on-line capabilities:

- A) Modification of instructions/statements;
- B) Deletion of erroneous code;
- C) Addition of new code;
- D) Rearrangement of code to change the logic of programs;

- E) Indefinite retention of the code in the system in a form that is immediately accessible for the above editing functions and recompilation/reassembly;
- F) Easy accessibility of all sections of a procedure for modification.

The Language Editing Overlay (LEO) has been implemented to provide for all of the above. The method of performing these functions in LEO is discussed in the sections on editing and scanning.

#### 3.1.3.1 Editing

The editing capabilities of LEO which may be performed are replacement, deletion, insertion, and modification.

When the user wishes to begin editing, the BEGIN/END EDIT key is activated (see Figure 3-3 for overlay layout diagram). This puts the user in the editing mode until the BEGIN/END EDIT key is activated again.

When in the editing mode, the user has the option of either inserting, replacing, or deleting code in a previously written section of a CODAP, DOL, or FORTRAN program. When any one of these keys is activated, a message is displayed on the display console requesting the line number of the statement to be edited. The keys (and/or typewriter) of a language overlay (CODAP/DOL/FORTRAN) may be used to create a new coding line image. The user activates the ENTER key and the intended operation is executed. In constructing a new line it is not necessary to fill the remainder of the line with blanks or to write the line number. This is done automatically when the ENTER key is activated.



# RIGHT PROGRAM KEY OVERLAY

OVERLAY NR 2

OVERLAY NAME LEO

PREPARED BY

START	P-31		P-32	TURN ON LIGHT	P-33	TURN OFF LIGHT	P-34		P-35
	P-36		P-37		P-38	ENTER	P-39		P-40
BEGIN/END EDIT	P-41	DISPLAY LAST DECLARATIVE PAGE	P-42	DISPLAY LAST MAIN PAGE	P-43	NEGATIVE LAST EDIT OPERATION	P-44		P-45
INSERT LINE	P-46	REPLACE LINE	P-47	DELETE LINE	P-48		P-49	ASSEMBLY OUTPUT CONTROL	P-50
NEXT PAGE	P-51	PREVIOUS PAGE	P-52	SCAN FORWARD	P-53	SCAN BACKWARD	P-54		P-55
	P-56	DISPLAY TO WORK TAPE	P-57	CODING SHEETS TO WORK TAPE	P-58		P-59	LIST PRINT TAPE	P-60

3 - 3

After the ENTER key is pressed, the editing message is automatically updated and rewritten so that the user can perform the same operation on the next line in sequence by merely pressing the ENTER key again. Thus, for example, the user may delete six sequential lines by pressing DELETE LINE once, typing in the number of the first of the six lines, and then pressing ENTER six times.

At any time, the user can press either INSERT, REPLACE, or DELETE, perform an operation different from the previous one, or operate on a line other than the one indicated in the message, and then fill in the required parts of the display area.

The user can negate the last completed editing operation by pressing the NEGATE LAST EDIT OPERATION key at any time after any editing operation has been performed on a page and if the user in the interim has not caused a different page to be displayed. The page will be restored to the form it had just previous to that last editing operation. This negation is then considered the last editing operation and it in turn may be negated by the same key.

It is possible when inserting lines to overflow the capacity of the page that is being displayed. The overflow lines are temporarily saved, and placed on the next page when editing is finished. Any subsequent overflow is similarly carried from page to page. The procedure is automatic, requiring no user intervention. Any subsequent viewing of pages shows the new form.

### 3.1.3.2 Code Scanning

Accessibility to all sections of a procedure is provided by the page scanning keys of LEO. These keys permit the viewing and possible alteration of any of the lines of code previously written. NEXT PAGE and PREVIOUS PAGE will display on the console the page that

preceeds or follows the page that is currently displayed on the screen. SCAN FORWARD and SCAN BACKWARD display successions of pages in a continual forward or backward movement of about 3 second intervals per page. DISPLAY LAST DECLARATIVE and DISPLAY LAST MAIN PAGE display the last Declarative or Main page display coded by the user. These keys will most often be used in the initial coding of a program when a user has left the main flow of coding to correct previously coded display pages and requires a return to the last page being worked on.

### 3.1.3.3 Access and Storage of Programs

Since programs may be changed and recompiled many times after they are originally written, provision must be made for saving them within the system so they may be accessed or altered easily. The programs are originally written through the FORTRAN or CODAP overlays as line images on display pages. They are saved within the system as a multi-paged display on the DOCUS library tape.

In order to describe these keys and how they fill the needs of a user, the process of initially writing a program and bringing it to production basis will be described.

The program is first written through the use of the FORTRAN or CODAP overlays. The editing and code scanning features of LEO can be used at this time to facilitate the process. The program is then compiled/assembled through the COMPILE/ASSEMBLE key of the language overlay.

After compilation of a program(s), the user may view a listing by using the LIST PRINT TAPE key on LEO. Logic or diagnostic errors can be immediately corrected and recompilation accomplished by

use of the CODING SHEETS TO WORK TAPE key. The coding sheet displays of the programs are brought back to the work area for viewing and modification of the programs accomplished with the various keys of LEO. This process may be repeated as often as desired. The ASSEMBLY OUTPUT CONTROL key is used when no immediate changes to the program are required. With this key the coding sheet displays of the program and/or the displays of the listing on the DOCUS library tape are saved. At a future time the DISPLAY TO WORK TAPE key may be used to view or restore either of these displays to the work area with subsequent editing of the coding sheets for recompilation to repeat the entire process. (The keys DISPLAY TO WORK TAPE, ASSEMBLY OUTPUT CONTROL, LIST PRINT TAPE, TURN ON LIGHT, and TURN OFF LIGHT are also on the PIL overlay and are more fully described in the PIL section of this paper.)

If the coding sheet displays are saved after compilation/assembly, the programs may be altered in the future. No hard copy or any other form of the program is ever required.

## 3.2 PROGRAMMING LANGUAGES

The programming languages which are implemented under DOCUS are CODAP and FORTRAN. A Display Oriented Language (DOL) is partially complete. These languages differ from the applications languages in that they may be used to solve many classes of problems as opposed to a specific application.

### 3.2.1 Display Oriented Language (DOL)

#### 3.2.1.1 Introduction

In the other sections of this report, the terms, man-machine system and man-machine communication, are used in describing or characterizing DOCUS. In this context, when we speak of the "machine", we are not merely talking about computer hardware. We, instead, are referring to an "extended machine", that is, hardware augmented by software. For the machine to be of assistance to the man in "augmenting his reasoning" or "amplifying his intelligence", rather sophisticated software must be coupled with the computer equipment. This software serves various functions. Some of these are:

- A) It may be dedicated and oriented to a specific application or problem;
- B) It may provide the tools and building blocks for constructing additional software;
- C) It may act as the "voice" of the machine in carrying on the necessary dialogue with the man.

It was recognized at the inception of the project that an appropriate language was needed for efficient implementation of the software needed for the man-machine dialogue. Early attempts to define this language produced the set of statements described in the Interim Technical Documentary Report of April, 1965. This form of the Display Oriented Language (DOL) proved to be too highly oriented towards the special characteristics of the Bunker-Ramo 85 display console and not adequate for the task of implementing key functions without the inclusion of large segments of machine code. During the last quarter, the effort to specify a suitable language has resumed and intensified bolstered by the experience gained in the implementation of several DOCUS overlays. Specification is not yet complete and only highlights of the results will be reported here. A subsequent report will provide complete specifications of the language and serve as a user's reference manual.

#### 3.2.1.2 Objectives

The objectives and goals of the development effort on the Display Oriented Language are:

- A) Provide a suitable language for stating the display oriented functions of the required software;
- B) Provide a collection of general purpose utility programs for manipulating display consoles;
- C) Provide a translator (compiler, macro assembler, preprocessor, or whatever) to transform DOL statements into machine executable code;

- D) Allow intermixing of other languages (to whatever extent feasible and desirable) for stating the data processing and computational functions of the required software;
- E) To be as equipment independent as possible in concept, design, and implementation;
- F) To be of assistance not only in developing applications software but in developing DOCUS system's software as well.

#### 3.2.1.3 Generalization of the Functions in Man-Machine Communication

In order to achieve as large a degree of equipment independence as possible, a generalization of the functions or actions which can be taken by man and machine was made.

The console user can perform one or more of the following functions in communicating with the machine:

- A) Interrupt - The user can take some action to interrupt the normal sequencing of the computer or to set a "service needed" flag signalling that the console has information for the computer.
- B) Select - The user can make a selection either by activating a console selector device or by designating an element of the display on the CRT scope.
- C) Enter Message - The user can enter by an alphanumeric keyboard and other manual input devices messages for transmission to the computer.

The machine in turn communicates with the man by performing one or more of the following functions:

- A) Set Indicators - Indicators are used to inform the user of the status of the problem, to guide him in the selection of the next problem step, or to inhibit interrupts (by hardware or software means).
- B) Transmit Message to Display on Scope - Display message may consist of "blanks" to be filled in, selection list, informative message, or a combination of these.

#### 3.2.1.4 Generalization of a Display Console

Analysis of the above functions and of the hardware characteristics of various display consoles currently available led to a set of characteristics or hardware features of a generalized console.

##### 3.2.1.4.1 Display Console Features

A generalized display console will be assumed to have the following features:

- A) Indicators - Visual or aural devices having two states, "on" or "off".
- B) Selectors - Switch devices which when activated cause the device number to be registered and/or transmitted to the computer.



- C) **Attention Requests** - Devices which can demand or request attention from the computer.
- D) **Attention Suppressors** - Devices which prevent interrupts from operating or which indicate they are to be ignored.
- E) **CRT Scope** - With symbol display and with line vector capability.
- F) **Alphanumeric Keyboard**.
- G) **Pointers** - For designating a selected display element - a symbol or a line segment.
- H) **Various Manual Controls** - Used to enter data (symbols or lines) on the scope and to manipulate (move, copy, delete, etc.) data elements on the scope.

A specific console may have various quantities of these features. Certain features may be missing. Certain features may be combined.

#### 3.2.1.4.2 The BR85 Console Measured Against this Generic Console

The BR85 console considered from this generalized viewpoint has:

87 Indicators	60 function key lights,
	25 status lights, a blinking
	error light, and an audible
	alarm

62 Selectors	60 function keys, 2 overlays
63 Attention Requests	the 62 selectors above (note the combined functions in a single device) plus a light pen
62 Attention Suppressors	2 hardware inhibitors (status lights 01 and 02), plus 60 function key lights interpreted by software to inhibit function key interrupts
1 CRT Scope	with 64 unique symbols and with line drawing capability
1 Alphanumeric Keyboard	
2 Pointers	a light pen and a crosshair cursor
Plus various manual controls which are not controllable by the computer	

### 3.2.1.5 Proposed Realization of DOL Capability

The first realization of DOL capability will be by means of a preprocessor which will accept as input a mixture of DOL and FORTRAN statements, transform these into pure FORTRAN statements, and drive directly an existing FORTRAN compiler to produce executable machine code. Figure 3-4 shows the schematic for this preprocessor/compiler coupling. Note that the user need not be aware of the preprocessor and that the intermediate results feed directly into the FORTRAN compiler without further intervention of the user. Most executable DOL statements will be transformed into CALLs on DOL subroutines stored in a library. Figure 3-5 shows run time loading flow.

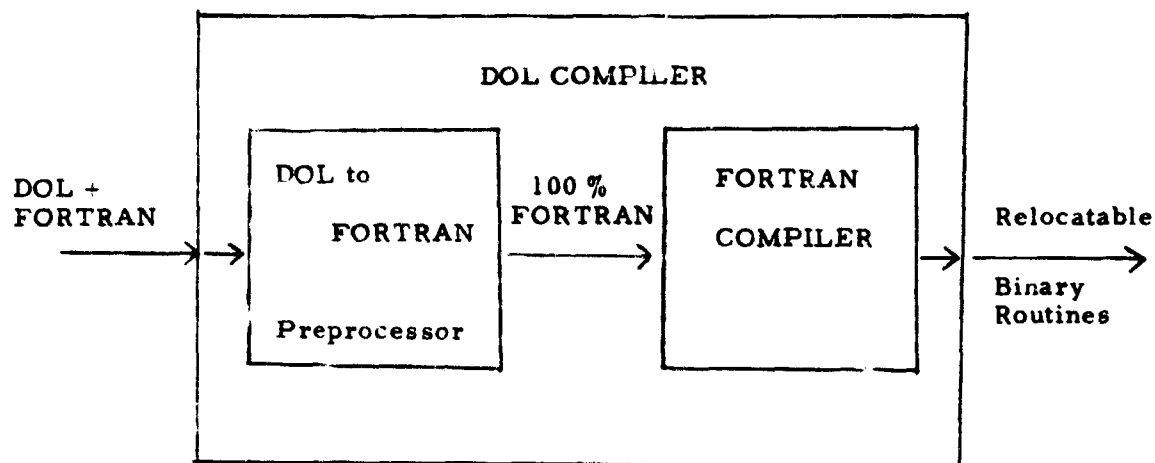


Figure 3-4  
Schematic of DOL Compiler

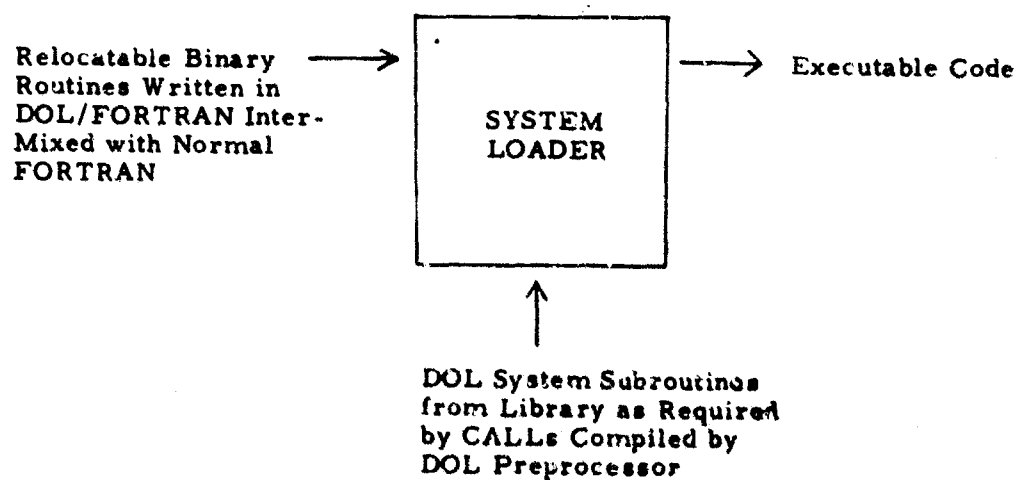


Figure 3-5  
Run Time Loading Flow of DOL Subroutines

### 3.2.1.6 DOL Statement Formats

DOL statements will adhere to FORTRAN conventions. A FORTRAN-type label may be given in columns 1-5 for executable DOL statements. The body of the statement begins in column 7 and is free form as in FORTRAN. The general form of a DOL statement is:

Operation (Operand List with Comma Separators)

Example:

TURN ON LIGHT (ENTER)

### 3.2.1.7 DOL Statement Classes

DOL statements may be classified as follows:

#### A) Declaratives or Specifications

- ... Used to name console features or display elements
- ... Used to define groups of console features
- ... Used to specify sizes, modes, and other attributes of display elements

#### B) Console Communications

- ... Used to read or write the contents of a display console memory
- ... Used to interrogate console registers
- ... Used to transmit indicator status to console

**C) Textual Display Generation and Manipulation**

... Used to manipulate in various ways the two-dimensional strings of characters making up a textual display

... Used to dissect a user response and extract message components

**D) Graphical Display Generation and Manipulation**

... Used to generate and manipulate graphical displays composed of line segments

**D) System Linkages and Environmental**

... Used to store into and retrieve from the library

... Used to communicate with the executive control

**3.2.1.8 Typical DOL Statements**

A complete description of all allowable DOL statements is not yet available but will be published subsequently in a DOL User's Manual. The following list of typical DOL statements is given here to provide the reader with a sampling of DOL capability.

**A) Declaratives or Specifications.**

- 1) **NAME LIGHT (name, number)** - Assigns the name given to the numbered light specified. Subsequently the light may be referenced by this name.

- 2) **DEFINE LIGHT GROUP** (name, N, light name or number, light name or number, etc.) - The N lights specified by name or number will henceforth be considered as an entity and may be referenced by the name assigned.

**B) Console Communications**

- 1) **WRITE CONSOLE** (name of the table for the display message) - The contents of a display message buffer is transmitted under direction of a control table to the console currently active.
- 2) **TURN LIGHT OFF** (name or number of a single light or name of a light group) - The light (or lights) specified has its status bit in the light mask of the currently active console set to zero ("off").

**C) Textual Display Generation**

- 1) **MOVE CHARACTER MATRIX TO DISPLAY MESSAGE** (name of control table for character matrix, name of control table for display message buffer, mode) - This moves text from a line/column array of characters into a display message buffer reformatting into the message form required by the currently active console. Two modes are available - **CLEAR**, which destroys any previous information in the display message buffer and **SUPER** (for Superimpose), which adds the new information to any previously stored in the display message buffer.

- 2) **SET BEAM POSITION** (name of control table for display message buffer, y coordinate, x coordinate, mode) - This sets the beam position or the next scope position at which information is to be displayed. Setting of the beam position parameter in the display message control table and generating and storage in the display message of appropriate positioning control bytes is accomplished. Mode indicates if coordinates are in scope units or in line/column units.

**D) Graphical Display Generation and Manipulation**

- 1) **EXTEND LINE** (name of a control table for a display message buffer, x coordinate, y coordinate) - This causes the control bytes for drawing a line vector from the previous terminal point of a line to the coordinates given to be generated and stored in the display message buffer.
- 2) **MOVE LINE ARRAY TO DISPLAY MESSAGE** (name of control table for line array, name of control table of display message buffer, mode) - This statement is similar to the **MOVE CHARACTER MATRIX TO DISPLAY MESSAGE** with the one difference that the information to be transferred is an array of points describing a set of one or more lines.

**E) System Linkage and Environmental**

**SWITCH ON CONSOLE RESPONSE**

$(R_1, S_1, \dots, R_n, S_n)$

This causes a return from a function program to the Executive voluntarily suspending the program until one of the responses ( $R_i$ ) is made by the console user. The Executive turns on indicators as appropriate to inform the console user of the response choices available to him. When response  $R_i$  is made, the Executive reactivates the function program at the statement  $S_i$  corresponding to  $R_i$ .



### 3.2.2 FORTRAN Language Overlay

#### 3.2.2.1 Introduction

Higher order languages provide single statement procedures which when compiled generate many sequential programming steps. Compiler languages are therefore a convenient tool for the programmer/analyst in that they enable him to define a problem in a format which is closer to the problem statement in English. The programmer is thus able to remain closer to the problem and give less attention to pure programming details. Therefore it was felt that a compiler oriented language should be implemented within DOCUS, serving as a major procedure implementation tool.

FORTRAN remains the most widely used and understood compiler language. It contains most of the assets of compiler languages in general in a brief, but flexible, repertoire of procedures. It is the one compiler which has been implemented for almost all available computers, and while various FORTRAN compilers are not totally compatible, they still represent the closest approximation to machine independence currently available. For these reasons FORTRAN is probably the most valuable internal tool for problem solving.

The purpose of the FORTRAN overlay is to allow on-line coding and compilation of FORTRAN language programs from the BR85 console. This overlay also allows the user to easily enter programs into the overall DOCUS system of key programs and subroutines and to modify these programs at any future date. These last two functions are done in conjunction with the PIL and LEO overlays.

The basic operation involves a user sitting at the BR85 console, writing a FORTRAN program, and using the display console as a coding pad with the typewriter keyboard and function keys as pencil and eraser.

FORTTRAN statements are retained in the system in original source language permitting subsequent editing by LEO. The output from a FORTRAN compilation is in a format that is accessible to other overlays in the DOCUS system. Thus the basic language overlays (CODAP and FORTRAN) are vehicles for entering programs into the system.

### 3.2.2.2 Coding Aids

The FORTRAN overlay provides several aids in writing a program on the CRT screen. Most compilers require declaratives to appear as the first statements of a program and will not accept mixed declarative and executable statements. When coding a program, it is usually not apparent at the outset exactly what storage allocations, constants, equivalences, etc., will eventually be needed. The solution used by most programmers is to have at least two sets of coding sheets. One set contains the main sequence for coding executable statements. The other set is used for declarative statements.

An important feature of the FORTRAN overlay is the capability of writing declarative statements in sequence with executable statements. Executable statements are written in the usual manner. When the need for a declarative statement becomes apparent, the user writes that statement on the screen as the next statement in sequence. An indication is made to the overlay program that a non-executable statement has been written by pushing the DECLARATIVE STATEMENT key of the FORTRAN overlay. The Declarative Statement line image is then stored in a secondary page buffer. This facility provides the user with two current coding pages to interact upon, a Declarative page and a Main page.

Other aids at the user's disposal include:

- A) Automatic updating of the line sequence number on each new statement (for ease in updating - see description of LEO);
- B) Automatic tabbing to correct position on display coding sheet;
- C) Automatic typing of some portions of statements. (Depression of function keys automatically writes the fixed portion of the statement on the display leaving the user to supply only the variable information.)

A coding sheet background is projected as part of the display at all times the user is writing his program. It delineates the statement number field, statement field, and line sequence number field. The area below the bottom horizontal line is used for messages from the various key programs to the user. Figure 3-6 shows the first page of a new program. All options normally available on a 7/9 FTN control card under CDC's 1604 FORTRAN 63 may be indicated on the control card on the screen - with the exception of the input unit which must be logical tape 5 and the Load-And-Go tape which, if requested, must be logical tape 56 (standard scratch unit 01).

#### 3.2.2.3 Keys

There are two types of keys on this overlay - Statement Writing keys and Control keys. The Statement Writing keys are used to write the fixed portions of FORTRAN statements on the CRT screen (such as "DIMENSION", "WRITE OUTPUT TAPE", etc.). The display marker is left positioned so that the user may type in the variable portion of the

STATE- MENT NO.	FORTRAN STATEMENT	LINE NO.
7		
7/9 FTN,	I = 5, ↓	
UPPER CASE MODE DECLARATIVE PAGE      UPPER SHIFT MODE		

Display of FORTRAN Coding Sheet  
Figure 3-6

statement using the console keyboard. Each Statement Writing key may cause either of the two statements indicated on it (see Figure 3-7 overlay layout diagram) to be displayed, depending on the current shift mode - upper shift mode for the statement indicated on the upper half of the key and lower shift mode for the statements on the lower portion of the key. The SHIFT key controls the current shift mode. Pressing the SHIFT key changes the shift mode to the opposite of what it was. Status lights and a message on the screen indicate the current shift mode.

- A) TAB Key - This key moves the marker to either Column 1 or Column 7 so that the user may begin coding in that column.
- B) PROGRAM, FUNCTION, or SUBROUTINE Keys - These keys indicate that the user wishes to start a new program in his series. Two new pages are created (Main and Declarative pages - see description below) for each new program. A Declarative page is always displayed at this initial point in the FORTRAN coding.
- C) DECLARATIVE and MAIN STATEMENT Keys - These two keys allow the user to intermix executable and non-executable FORTRAN statements. Two displays (pages) are kept active - one for the executable statements (Main Statements) and the other for non-executable statements (Declarative Statements). At the completion of each statement the user must signify which FORTRAN statement he has just completed.
  - 1) DECLARATIVE STATEMENT Key - A Declarative Page is the first display to appear after the initial program or subroutine naming definition. The user now

LEFT PROGRAM KEY OVERLAY

OVERLAY NR  
OVERLAY NAME FORTRAN

PREPARED BY

START	P-1	DIMENSION SENSE LIGHT	P-2	WRITE TAPE WRITE	P-3	READ TAPE READ	P-4	MAIN STATE- MENT	P-5
PROGRAM END	P-6	TYPE IF SENSE LIGHT	P-7	WRITE OUT PUT TAPE BUFFER OUT	P-8	READ IN- PUT TAPE BUFFER IN	P-9	NEW PAGE	P-10
FUNCTION RETURN	P-11	DATA IF SENSE SWITCH	P-12	PUNCH PRINT	P-13	PAUSE IF	P-14	CONTINUE DO	P-15
SUB-ROUTINE ENTRY	P-16	EQUIVA- LENCE IF DIVIDE FAULT	P-17	ENCODE DECODE	P-18	STOP CALL	P-19	ASSIGN GOTO	P-20
COMPILE	P-21	EXTERNAL IF EXPO- NENT FAULT	P-22	IF IO CHECK BACKSPACE	P-23	IF UNIT REWIND	P-24	TAB	P-25
DECLA- RATIVE STATE- MENT	P-26	COMMON IF OVER- FLOW FAULT	P-27	IF EOT END FILE	P-28	FORMAT	P-29	SHIFT	P-30

Figure 3-7

inputs non-executable statements by using a combination of function keys and the console typewriter. Each statement is terminated with the depression of the DECLARATIVE STATEMENT key. This action affixes a sequence number to the statement and is assigned as a non-executable type statement. The display marker is moved to column 7 of the display coding sheet for the next statement. (If the user desires a statement number or a comment - depression of the TAB key will place the marker in column 1 of the display coding sheet.)

- 2) MAIN STATEMENT Key - This key is activated after the user has inputted (via keys and/or typewriter) a Main Statement. Depression of this key affixes a line sequence number and assigns the statement as an executable type. The first time this key is activated the display changes from a Declarative Page (first page is always a Declarative Page) to a Main Page display. From this point until the "END" statement is inputted the Main Page display is projected. To input a Declarative statement the user activates the DECLARATIVE STATEMENT key and the last statement inputted is erased from the console display and assigned as a non-executable type of statement.

- D) **NEW PAGE Key** - Users often wish to maintain one series of statements - such as a DO loop - on the same page. This may be accomplished by pressing this key. Further, a new page is automatically displayed as soon as a user completes the present page.
- E) **COMPILE Key** - This key transforms all the statements previously written on the CRT screen into a form acceptable to the FORTRAN 63 compiler and has them compiled. Prior to compilation the user is given an opportunity to specify the storage media of the listing output to produce it on either the printer for hard copy or on a tape for later viewing on the CRT screen (by using the **PRINT OUTPUT TAPE** key on the **PIL** or **LEO** overlays). The user may also specify whether the original displays of statements are to be saved. If so they may be retrieved for editing and recompilation immediately after compilation by using the **RETRIEVE CODING SHEETS** key on the **LEO** overlay or he may put them on the library tape for later access using the **ASSEMBLY OUTPUT** key on the **PIL** and **LEO** overlays. After compilation, the **COMPILE** key sends a message informing the user as to the outcome of the compilation.

The FORTRAN overlay used in conjunction with LEO is a powerful tool that enables the user to employ all of the advantages of coding on line without loss of any of the advantages available to the off line FORTRAN programmer.



### 3.2.3 CODAP Language Overlay

#### 3.2.3.1 Introduction

Whereas compilers are programmer or problem oriented, assembly languages are computer oriented. The instructions or operations in an assembly language exist in a one-to-one correspondence with the hardware functions of the computer. These elemental operations, represented by short mnemonic codes for programmer convenience, make problem definition possible in the most direct form, and in so doing require the least number of steps. Typically problems coded in this manner are economical in expenditure of the computer facilities.

The assembler language for the CDC 1604B, CODAP, has been implemented as one of the programming languages. It is a flexible tool suitable for any type of programming application. When core allocation is critical, CODAP can be used to obtain maximum efficiency by the experienced programmer.

The CODAP overlay as with the FORTRAN overlay allows the user an on-line programming capability using the CODAP symbolic language. Due to the nature of an on-line system, the user is allowed to:

- A) Assemble a program on completion of coding;
- B) Test the program;
- C) Enter data at execution time by the use of displays;
- D) Save source and object programs on the library;
- E) Have the results of the program displayed;
- F) Modify the program on line.

The CODAP overlay is an application language and was designed to be used in conjunction with an executive language, PIL. Indeed, it is PIL (and LEO) which allows the user the capabilities in D, E, and F above.

The CODAP overlay is designed primarily as a coding tutor with the majority of the keys devoted to describing various facets of the CODAP language. Coding is performed through the use of the typewriter. Thus, an on-line reference manual and a comprehensive source of utility routines are provided for the user. By having the instruction and utility routine descriptions readily available, less experienced programmers are able to improve coding competency, thereby increasing the capabilities the user has in the DOCUS system.

The keys for this overlay have been segmented into four:

- A) Instruction Tutorial Keys
- B) Subroutine - I/O Tutorial Keys
- C) Coding Keys
- D) Control Keys

#### 3.2.3.2 Instruction Tutorial Keys

The Instruction Tutorial keys list the mnemonics and machine code as a function of specific categories. That is, the CODAP instruction set is divided up by category into groups. Each group is assigned a key. When the key associated with a group is pressed, the instructions in this group along with a brief description of each instruction are displayed on the CRT. Figure 3-8 lists the Instruction Tutorial keys and the instructions associated with each key. Figure 3-9 is a facsimile of the display console when the DATA TRANSMISSION key is activated.

KEY NAME	INSTRUCTIONS
DATA TRANSMISSION	LDA, LAC, LDQ, LQC, STA, STQ
INDEX INSTRUCTIONS	ENI, INI, LIU, LIL, SIU, SIL, ISK, IJ, P
SHIFT INSTRUCTIONS	ARS, QRS, LRS, ALS, QLS, LLS
ADDRESS MODIFICATIONS	SAU, SAL
ARITHMETIC INSTRUCTIONS	ADD, SUB, MUI, DVI, MUF, DVF
FLOATING POINT INSTRUCTIONS	FAD, FSB, FMU, FDV
SCALE INSTRUCTIONS	SCA, SCQ
AQ INSTRUCTIONS	INA, ENQ, ENA
JUMPS AND STOPS	AJP, QJP, SLJ, SLS
STORAGE TEST INSTRUCTIONS	SSK, SSH
LOGICAL INSTRUCTIONS	SST, SCM, SCL, SSU, LDL, ADL, SBL, STL
SEARCH INSTRUCTIONS	EQS, THS, MEQ, MTH
REPLACE INSTRUCTIONS	RAD, RSB, RAO, RSO
TRANSFER INSTRUCTIONS	INT, OUT

Figure 3-8

LDA BM	(N)	REPLACE A
LAC BM	(N)'	REPLACE A
LDQ BM	(N)	REPLACE Q
LQC BM	(N)'	REPLACE Q
STA BM	(A)	REPLACE M
STQ BM	(Q)	REPLACE M

LEGEND:

B = Index Register	( ) = Contents of
N = M + (B)	' = Complement
M = Address	

Figure 3-9

### 3. 2. 3. 3 Subroutine I/O Keys

The Subroutine -I/O keys provide descriptions of commonly used utility routines (such as the CO-OP Monitor routines READ\*, WRITE\*) and the Input-Output codes used with the 1604 peripherals. Figure 3-10 is a facsimile when the Sense EXF. key is activated.

### 3. 2. 3. 4 Coding Keys

Each coding key is associated with a commonly used CO-OP Monitor subroutine (READ\*, WRITE\*, BCDBN\*, MEMREC\*, REMOVE\*, and SELECT\*). Upon pressing one of these keys, the calling sequence used for the appropriate routine appears on the screen (see Figure 3-11).

### 3. 2. 3. 5 Control Keys

The Control or Editing keys permit the user to tab to specific coding columns, create new pages, declare various statements as remote (constants) and assemble the program.

Coding of a CODAP program is achieved through the use of the console typewriter. The BR85 console has the facility to maintain two active displays within its memory, one display is used as a running CODAP coding sheet, while the other is used to display the tutorial information. The user has the responsibility to shift the console when the need arises (using a console selection key) from one type of display to another. The CODAP overlay (Figure 3-12) used in conjunction with the Executive overlay (PIL) and LEO allows the user to edit a CODAP program, incorporate the assembled CODAP object program into the DOCUS system, and perform other executive tasks.

## SENSE

Sense instructions sense the condition of an external equipment or the internal conditions of the computer and will execute a full exit or half exit depending on the presence or absence of the condition.

Instruction Format: 74 7 XXXXX

Where XXXXX is an External Function Code (EXF)

When used in the upper instruction position of a word and the condition tested is not present, the next instruction to be executed is in the lower half of the word. If the condition tested was not present and the sense instruction is in the lower half of the word, the instruction exists on itself until the condition exists.

- \* 1604 -A EXF Codes
- \* 1612 EXF Codes
- \* Console Equipment Codes
- \* 1615 Function Codes

For a list of the EXF SENSE codes associated with the 1604-A, Console, or the 1615, light-gun the appropriate asterisk.

Figure 3-10



# LEFT PROGRAM KEY OVERLAY

OVERLAY NR \_\_\_\_\_

OVERLAY NAME \_\_\_\_\_ CODAP \_\_\_\_\_

PREPARED BY \_\_\_\_\_

START	P-1	INDEX INSTRUCTIONS	P-2	SHIFT INSTRUCTIONS	P-3	BCDBN*	P-4	READ*	P-5
SENSE EXF CODES	P-6	SELECT EXF CODES	P-7	ACTIVATE EXF CODES	P-8	MEMREC*	P-9	WRITE*	P-10
ADDRESS MODIFICATIONS	P-11	ARITHMETIC INSTRUCTIONS	P-12	FLOATING POINT INSTRUCTIONS	P-13	REMOVE*	P-14	SELECT*	P-15
SCALE INSTRUCTIONS	P-16	A Q INSTRUCTIONS	P-17	JUMPS AND STOPS	P-18	NEW PAGE	P-19	DECLARATIVE STATEMENT	P-20
STORAGE TEST INSTRUCTIONS	P-21	DATA TRANSMISSION INSTRUCTIONS	P-22	TRANSFER INSTRUCTIONS	P-23	READ* / WRITE* TUTORIAL	P-24	MAIN STATEMENT	P-25
LOGICAL INSTRUCTIONS	P-26	SEARCH INSTRUCTIONS	P-27	REPLACE INSTRUCTIONS	P-28	COMPILE	P-29	TAB	P-30

Figure 3-12



### 3.3 APPLICATION LANGUAGES

Application Language overlays are those overlays which facilitate the solution of a specific class of problems and impose no coding requirements. Rather what is required is a complete man/machine stimulus/response to achieve the user's ends. The application overlays, File Management, and Text Manipulation, have been implemented in this manner, designed to serve the analyst's requirements.

#### 3.3.1 File Management

##### 3.3.1.1 Introduction

The ability to absorb information, retain it, and recall it when necessary are vital attributes of the decision making process. The individual analyst does a fair job on the above when the information involved is relatively finite and the time factor between absorption and recalling for use is minimal. However in this era of the information explosion further advances in computer technology are required to record, store, and transmit information.

The DOCUS File Management overlay is a step in this direction. This overlay can be regarded as an information storage media to which access is granted as needed, with the human being still retaining decision making prerogatives.

The File Management overlay provides a user with the capability for on-line storage and retrieval of a data base. Using the display console in conjunction with the console keys, a dialogue is developed between the user and the computer. This dialogue affords the user the capability of defining, modifying, and querying data fields to suit particular needs.

With proper utilization of the unique capabilities of a CRT, desired operating environment can be achieved by presenting the user with pertinent information about the data base at each step of the dialogue. With this capability, the user need not be a systems expert nor even knowledgeable about the physical structure of the particular data base to be used. The system guides him step by step, through each phase of any desired operation. For example, during the File Definition operation, a formatted display is presented for entering file structure information. From this information, the system now constructs detailed displays of the file, which are available in other phases to aid the user in formulating updates and queries. The user is guided by a complete set of tutorials describing the actions that can be taken to execute his problem. Figure 3-6 is a description of the File Management Overlay.

#### 3.3.1.2 File Definition Phase

The File Definition phase consists of the steps of naming, describing, and defining the physical structure of the file. The basic level of a file is a field. A group of fields which uniquely describe an entity are grouped in a record. Records of a similar form or structure are grouped together to form files. It is clear, therefore, that in the File Definition phase, the three items that must be described by the user are: 1) file, 2) record, and 3) field.

The file name (which is the first step in file definition) must be unique. Therefore, the system has been designed to perform checks to prevent files bearing identical names from being incorporated into the base data. The user is informed of identical naming and presented another opportunity to re-name the file with some unique alphanumeric combination. The system next provides the user with the capability of entering a brief file description at this time (Figure 3-7). This serves as an analyst's aid in retrieving files when no other descriptive material is available.

LEFT PROGRAM KEY OVERLAY

OVERLAY NR 19  
OVERLAY NAME FILE MANAGEMENT

PREPARED BY \_\_\_\_\_

ACTIVATE OVERLAY	P-1	ENTER	P-2	P-3		P-4		P-5
FINISHED	P-6		P-7	P-8		P-9		P-10
	P-11	FILE DEFINITION	P-12	P-13		DELETE RECORD		P-15
	P-16	ADD RECORD	P-17	P-18	FILE MAINTENANCE	P-19	LESS THAN	P-20
EQUAL TO	P-21	NOT EQUAL TO	P-22	P-23		QUERY	FILE KNOWN	P-25
FILE UNKNOWN	P-26		P-27	P-28	CRT OUTPUT	PRINTER OUTPUT		P-30

Figure 3-6

## FILE DEFINITION DISPLAYS

The diagram illustrates two separate input areas for file definition, each enclosed in a rounded rectangular frame. The top frame contains the text 'FILE NAME' followed by a horizontal dashed line for input, with a small upward-pointing triangle positioned below the first dashed line segment. The bottom frame contains the text 'GIVE A BRIEF DESCRIPTION OF FILE' followed by two horizontal dashed lines for input, with a small upward-pointing triangle positioned below the first dashed line segment.

FILE NAME \_\_\_\_\_  
△

GIVE A BRIEF DESCRIPTION OF FILE  
\_\_\_\_\_  
\_\_\_\_\_

Figure 3-7

Once the file has been named, and a description given, the structure of the records incorporated in the file must be described as well as the fields which make up the record. Both are accomplished by presenting the user with a display (Figure 3-8), which allows for the specification of the number of fields desired; the field name; the mode (alpha or numeric); and the length of each field. Each display can contain up to 25 field definitions, which in turn, delineates a record definition. Once the user signifies completion of field descriptions, the defined fields are entered into the system. The record definition display is stored and presented to the user during any future file maintenance. The record display enhances the probability of correctly adding, deleting, or modifying data in the data base (See Figure 3-9).

A data dictionary is internally generated by the system for use in other phases. All fields of a file are packed conserving space on the peripheral device and minimizing Input and Output operations.

#### 3.3.1.3 File Maintenance Phase

The File Maintenance phase accords the user the ability to add and/or delete records to establish files. The user must define the file he wishes to modify. Tutorial guidance is provided to the user if the name of the file is not known by the use of a description display on the CRT. This display lists all files by name with corresponding descriptions of the nature of each file. The user can designate the proper file by use of the light pen. Once the name of the file has been communicated to the system, all pertinent information with respect to format and field structure is made available to the user for manipulation. The ADD RECORD and the DELETE RECORD functions permit the user to select the next required action. If the ADD RECORD function is selected, the record definition display is presented

FIELD DEFINITION DISPLAY

	NAME	MODE	LEN
FIELD 1	— — — — —	—	—
FIELD 2	— — — — —	—	—
FIELD 3	— — — — —	—	—
FIELD 4	— — — — —	—	—
FIELD 5	— — — — —	—	—
FIELD 6	— — — — —	—	—
FIELD 7	— — — — —	—	—
FIELD 8	— — — — —	—	—
FIELD 9	— — — — —	—	—
FIELD 10	— — — — —	—	—
FIELD 11	— — — — —	—	—
FIELD 12	— — — — —	—	—

Figure 3-8

# FILE RECORD FORMAT

<u>NAME</u>	<u>MODE</u>	<u>LEN</u>	
TYPE	*A	8	<u>▽</u> _ _ _ _ _
BRAND	*A	4	_ _ _ _ _
DESIG	*N	3	_ _ _ _ _
LOCATION	*A	2	_ _ _ _ _
ON HAND	*N	8	_ _ _ _ _
ON ORDER	*N	8	_ _ _ _ _
ITEM COST	*N	6	_ _ _ _ _

Figure 3-9

on the CRT. The user types desired data via the console typewriter for this record and signals the program via the ENTER key when all desired data for this record has been entered. The record is then added to the file. The ADD RECORD and DELETE RECORD function keys are now reactivated to allow further file maintenance function.

In the DELETE RECORD function, the same format (as in the ADD RECORD function) is displayed, e. g. , a mask of the record format. The system instructs the user to light gun those fields which would uniquely identify the records to be deleted. After field selection is complete, the user is allowed to type in the criteria for each field. Upon finishing this type-in, the system is entered and all records are deleted from the file whose fields are identical to those of the type-in. If no records are found which contain the specified field values, the user is informed by a message on the CRT. Requests can be respecified by repressing the DELETE key function. If further maintenance action is required, the FILE MAINTENANCE key is activated and the user's options of maintenance are again available.

#### 3.3.1.4 Query Phase

The Query phase of the File Management system allows the user to ascertain those records which he desires to examine.

The user requests that a particular file be subject to query (the system can guide a user who does not know the name of the file). A display of all the fields in the file is presented. The user requests that certain threshold criteria be applied to the individual fields (less than, greater than, equal, not equal, by use of light pen for field selection and function keys for operators). The user designates the criteria desired, supplying a threshold value to test against; e. g. , if the user wants all records that contain for field A a value greater than 20, the greater than criteria is selected with an input value of 20.



Criteria can be placed on more than one field in a record. A search is now made of the data base. The Query language disregards those records in the data base that do not fulfill all the field criteria designated by the user. When the search has been completed the user is afforded the option of having a hard copy output of the selected data or an on-line display output (or both).

### 3.3.2 Text Manipulation

#### 3.3.2.1 Introduction

The storage, retrieval, and editing of "text", i.e., strings of alphanumeric characters, is an important part of most business, and professional activities.

- A) Many activities involve the production or revision of text of one sort or another, e.g., reports, memos, summaries, lists, tables, etc.
- B) Most activities involve at least the perusal of text. Often, large quantities must be scanned as rapidly as possible with the objective of locating essential details occupying a small portion of the text.
- C) Some activities involve research with text itself, e.g., machine translation, auto-abstracting, etc.

For all of the above applications, it is desirable to have a capability for:

- A) Entering a body of text into a computer/display system and storing in a form suitable for either immediate viewing or storage for viewing at a later time;
- B) Viewing and scanning through the text in an easily readable form on a suitable display device. The viewer should be able to control the rate and direction of scan;
- C) Tagging and identifying elements of special interest and importance;

- D) Editing - insertion, deletion, and alteration of text elements;
- E) Extraction of excerpts for serarate viewing, editing, and storing.

The Text Manipulation overlay is designed to meet these requirements.

#### 3.3.2.2 Capabilities

Text Manipulation Language provides a capability for displaying textual information, rolling it up or down on the display, and performing editing operations on the textual information.

- o The layout of keys for the Text Manipulation overlay is shown in Figure 3-10. The keys fall into four groups:
  - A) Text input/output.
  - B) Text rolling control.
  - C) Text operand keys, e.g., character, word, etc.
  - D) Text operation keys, e.g., insert, delete, etc.
- o Text is input from one or more input media, e.g., paper tape, cards, etc., by use of the input text key. It is converted to a standard form on magnetic tape.
- o The first page of text is then projected onto the CRT screen. At this point, the screen looks as shown in Figure 3-11.
- o The operator may then roll the text forward (up the screen) and backward (down the screen) in "continuous"

LEFT PROGRAM KEY OVERLAY

OVERLAY NR \_\_\_\_\_  
OVERLAY NAME TEXT MANIPULATION

PREPARED BY \_\_\_\_\_

INPUT TEXT	P-1	RESTART	P-2	UPDATE TEXT	P-3	OUTPUT TEXT	P-4	CURSOR CONTROL	P-5
ROLL FORWARD	P-6	ROLL BACKWARD	P-7	ROLL FASTER	P-8	ROLL SLOWER	P-9	STOP ROLLING	P-10
AT	P-11	CHARACTER	P-12	WORD	P-13	CLOSED EXPRESSION	P-14	CLAUSE	P-15
SENTENCE	P-16	PARAGRAPH	P-17	STRING	P-18		P-19		P-20
DELETE	P-21	INSERT	P-22	SUBSTITUTE	P-23	TRANSDUCER	P-24	TAG	P-25
SUPPRESS SPACES	P-26	EXTRACT	P-27	PARAGRAPH BREAK	P-28	PAGE BREAK	P-29	CANCEL LAST EDIT	P-30

Figure 3-10

18	{	Four score and seven years ago our fathers ...
Lines		...
of Text		...
3	{	... will little note nor long remember
Blank		_____
Lines		_____
Correction Area	{	▽
1 Line		_____
3		_____
Blank	{	_____
Lines		_____
Message Area		Press AT to edit. Press ROLL FORWARD or
2 Lines		ROLL BACKWARD to roll.

# INITIAL FORM OF TEXT

Figure 3-11

scroll fashion. The rate of roll may be varied both by use of appropriate function keys and by motion of the cursor.

At any point in the text the rolling may be stopped. Then, one or more editing operations may be applied to the portion of the text currently on the screen. The operations are applied to elements of text (operands) identified by the operator with the console's light pen and text corrections typed into a specially-provided correction area by the operator using the console typewriter.

- A) An operand may be any character, word, closed expression (e.g., parenthetical expression, expression in quotes), clause, sentence, paragraph or arbitrary string of characters. The operator selects an operand by pressing one of the operand keys, light-penning any character in the desired operand and then pressing the ENTER key. Thus, to select a given word in the text as an operand, press the key Word, light-pen any character in the given word and press ENTER.
- B) The operator enters a correction into the correction area by simply typing it on the console typewriter. It is not necessary to create a marker first, as one is provided by the system at the start of the correction area.

The operator may apply the following operations to the text:

- A) Insert either a correction or an operand after another operand (see Figure 3-12).

- B) Substitute either a correction or an operand for another operand (see Figures 3-13 and 3-14).
- C) Delete a specified operand (see Figure 3-15).
- D) Interchange (transpose) two operands (see Figure 3-16).
- E) Tag an operand so that it will blink thereafter whenever it is rolled onto the screen;
- F) Suppress a string of space-characters before or after the operand;
- G) Insert a paragraph break or page break after the specified operand. A paragraph break forces the operand following it to start a new line on the display screen. Similarly, a page break forces the next text element to start a new page, i.e., the specified operand becomes the last element on the current text page and remains so during any subsequent rolling until the page break is rolled off the screen;
- H) Put the specified operand on a list of "extracted text" which may then be displayed and rolled separately from the main body of text.

Note that in those operations involving two operands, the operands need not be of the same type, e.g., a sentence and a word may be interchanged. Note further that there is no restriction on the position of one operand relative to the other; it is only necessary that they both appear on the screen at the same time.

At any point in a given editing operation up to and including the point at which the editing operation is complete

. . . but in a sense we cannot . . .

4a. Before insertion of correction  
"larger" after operand "0".

. . . but in a larger sense we cannot . . .

4b. After insertion.

Figure 3-12

. . . shall not pass from the . . .

5a. Before substitution of correction  
"perish" for operand "pass"

. . . shall not perish from the . . .

5b. After substitution.

Figure 3-13



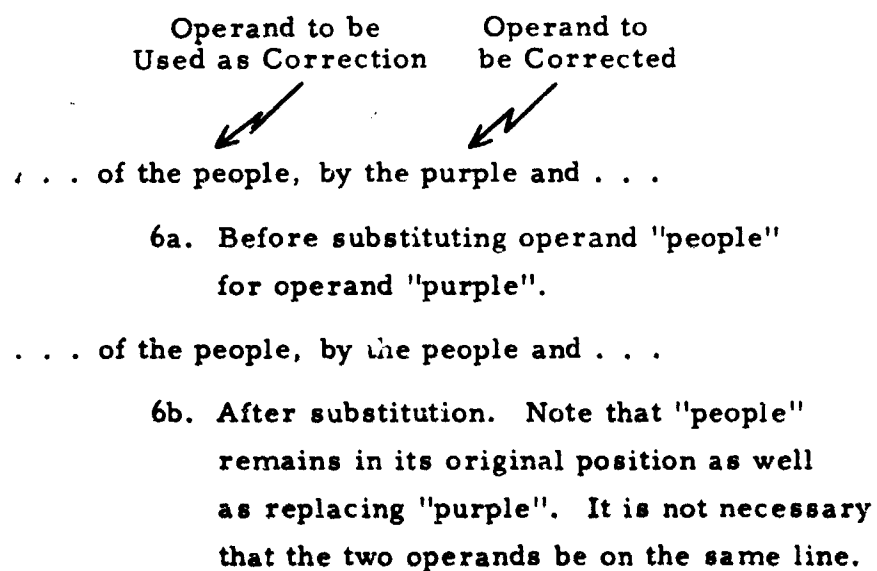


Figure 3-14

. . . for all of the people . . .


7a. Before deleting the string of  
characters, "all of".

. . . for the people . . .

7b. After deletion.

Figure 3-15

1st Operand      2nd Operand

  
. . . by the people, of the people . . .

8a. Before interchanging "by"  
and "of".

. . of the people, by the people . . .

8b. After interchange. The two  
operands need not be on the  
same line.

Figure 3-16

but before the next editing operation is begun, the current editing operation may be cancelled. Cancellation restores the system to the state it was in before the current editing operation was begun.

- o After any number of editing operations, the operator may choose to produce an updated standard input magnetic tape. This has no effect on how the text appears on the display screen. However, it can decrease the amount of time required to construct a text page and hence decrease console response time. More important, the input magnetic tape produced now includes all editing changes (these were previously contained in a core directory and change list); this magnetic tape can be saved and used as an input to the Text Manipulation system at a later time.
- o Finally, after any number of editing operations, the operator may output the text onto a specified output medium, e.g., punched paper tape, line printer, etc.

#### 3.3.2.3 Further Extensions of Overlay

The system allows specification of operands in two ways. All operands except "Strings" are specified by a) pressing an operand key, e.g., CHARACTER, WORD, etc., b) light-penning any character in the desired operand. By contrast, strings are specified by light-penning the first and the last characters in the string. So far as the system is concerned, most editing could be done entirely with strings, since clearly the string is the most general of the operand types. However, the capability specifying the other types

of operands provides a significant convenience because a) only one character need be light-gunned, not two; any character in the operand may be light-gunned, not a specific character. Advantage a) obviously reduces the time required to perform an editing operation since fewer actions need to be performed. Advantage b) is significant because many people have difficulty in light-gunning the character they want to light gun. With an improved light gun, on future display consoles, b) may be less important.

There are several possibilities for extension of this effort:

- A) Currently, if an operand is to be specified in an editing operation, the entire operand must be capable of being simultaneously displayed on the screen. This constraint limits operands to a length of less than 18 lines of text. Two operand editing operations must be capable of being simultaneously displayed on the screen. The current system could be extended to allow operands of arbitrary length and as widely separated in the text as desired.
- B) The concept of "tagging" an operand currently means only that the operand will thereafter blink whenever it is rolled onto the screen. The system could be extended to allow operands to be tagged by name, syntactic type, or other interpretation. The directory could then act as a data source for a higher language interpreter, e.g., for syntax analysis, mechanical translation, post-editing, or other interaction between

an automatic language processor, and a human being.

C) Currently, the text is viewed primarily as a sequential character string. Editing consists primarily of 1) inserting a new substring at some point in the initial string, 2) deleting a substring, 3) replacing a substring by a new one, and 4) rearranging the order of substrings in the existing text. There is very little capability for two-dimensional formatting of text either on the display screen or on a hard-copy output device like a line printer. Currently, the formatting consists of the following:

- 1) The text appears on the screen as 18 lines of up to 64 characters, each line under the preceding one.
- 2) If the end of a line occurs in the middle of a word, the word is not broken into two parts; instead the entire word is moved into the beginning of the next line.
- 3) A paragraph break can be placed in the text between two successive text operands. Then, the first operand will always terminate the line in which it occurs and the second operand will always begin a new line. The new line will be indented by at least one character position which will contain a stop code.
- 4) Similarly, a page break can be inserted between any two successive text operands.

Note that all the above are essentially sequential string manipulation operations. In an extension of the current system, other formatting features could be provided, including 1) column formatting, 2) right as well as left justification of each line, 3) underlining of passages, 4) specified conversions applied to suitably-identified text types, and 5) insets in the text to provide room for illustrations, etc. Moreover, all of the above could be applied to hard-copy output as well as to CRT displays.

## APPENDIX A

### Definitions

The following terms, used throughout this report, are defined as follows:

Cursor	Displayable crosshairs whose position may be manipulated by the CRT console operator and the location of which may be interrogated by the computer.
Marker	Downward pointing arrow which denotes the next position where a typed-in character will occur on the display console.
Function Key	A switch on the console which causes an interruption in the computer which, in turn, permits a string of machine instructions to be executed.
Key Program	A group of machine instructions which are executed when a function key is pressed. Usually a key program is an elementary function, e.g., add item to list or sine X.
Key Group	Several associated or related function keys.
Page	The information, text or lines, which is displayed on the CRT at one time.
Display	One or more pages.
Status Light	A light on the console which can be turned on or off under program control.

Light Pen      A hand-held photo-electric device used to point at information being displayed on the screen to identify it to the console or computer.

List Display - The following figure shows a typical list display. Each item is preceded by an asterisk. When an item is selected with the light gun, the asterisk changes to an X.

THIS IS AN EXAMPLE OF A LIST DISPLAY

```
*      CHICAGO
*      LOS ANGELES
*      MIAMI
*      NEW ORLEANS
*      NEW YORK
*      SEATTLE
```

Format Display - The format space into which data is entered by the use of the alpha-numeric keyboard is specified by underscores. The entry point for the first character is specified by:

THIS IS AN EXAMPLE OF A FORMAT DISPLAY

```
LAT   _ _ _ _ _
LONG  _ _ _ _ _
DATE  _ _ _ _ _
TIME  _ _ _ _ _
```



## APPENDIX B

### DOCUS Hardware Configuration

The system operates with the following equipment configuration:

1604B computer

6 tape units on channels 3 & 4

printer on channel 7

BR85 console on channels 5 & 6

card reader on channel 1

typewriter on channels 1 & 2

paper tape punch on channel 2

paper tape reader on channel 1

## APPENDIX C

### C.1 Operation Under DOCUS Executive

The DOCUS Executive performs the following initial functions on receiving control via the MCS card:

- A) It sets all key and status lights to off condition.
- B) It sets Status Light 1 to on condition.

The result of this action is the placing of the BR85 "on-line", permitting the BR85 operator to generate a 1604B interrupt by placing a plastic overlay on one of the keyboards.

The general hardware rule for generating an overlay interrupt message is: If an overlay is placed on a keyboard, an interrupt will be generated if and only if Status Light 1 is on and another overlay is on the other keyboard. Thus, the proper procedure for placing an overlay on, say, the right keyboard (assuming there is initially no overlay on either keyboard) is:

- A) Place any overlay on the left keyboard. No interrupt will be generated.
- B) Place the desired executive overlay on the right keyboard. An interrupt will be generated. The Executive accepts the interrupt message, records the number of the overlay and turns on the upper left-hand light on the right keyboard (R1). Note that the key programs associated with this right overlay are not loaded into memory from the library at this time.

If it is now desired to place a left overlay as well on the keyboard, remove the current left overlay and replace it with the desired overlay. A left overlay interrupt is generated, the overlay number recorded as with the right overlay and key light L1 turned on.

The operator is now ready to push the START key, R1, on the right overlay. Since both overlays are down, pressing R1 generates a key interrupt. The Executive accepts the key interrupt; loads all the key programs and subroutines associated with the overlay on the right keyboard; saves the light settings of the other (left) keyboard; turns off all keyboard lights and transfers control to the key program associated with key R1. This "START" key program for the overlay performs various initialization functions and returns control to the Executive. The Executive turns on an initial set of key lights on the right keyboard. This initial set varies from overlay to overlay; the initial set for overlay n is specified in the nth word of the 64-word light table in the DOCUS directory. The Executive then returns to the idle loop.

The operator may now press any one of the lighted keys on the right keyboard, say Rn. The Executive accepts the interrupt message, saves the left key light settings again, turns off all keyboard lights and transfers control to the corresponding key program. The key program executes its intended function and may also turn on one or more right keyboard lights to indicate the set of related key programs, one of which the operator may next actuate. When this is done control is returned to the Executive. If the key program has turned on a set of key lights, the return calling sequence normally instructs the Executive to leave the keyboard lights alone. The Executive follows these instructions and restores the left keyboard lights to the settings which existed before Rn was pressed. If key program Rn did not turn on any key lights, normally it instructs the Executive

on return to reset the initial key lights. The Executive will turn on the initial set on the right keyboard and restores the previous light settings on the left keyboard, as before. In either case, the Executive then returns to the idle loop.

The operator may execute a series of key programs in the right overlay, guided by the key light settings. If, at any time, the initial set of key lights is not set, the operator may return to the initial condition by merely pressing the START key light, R1. If R1 is itself not lit, the operator picks up the right overlay and places it down on the right keyboard again. This will cause light R1 to go on. The overlay is then reloaded from the library tape when R1 is pressed.

At any time, the operator may choose to depress L1 instead. When that occurs, key programs and subroutines for the left overlay are loaded, right overlay light settings are saved, all keyboard lights are turned off and control goes to the key program associated with L1. When this key program returns to the Executive, the Executive turns on an initial set of key lights for the left overlay, restores the light settings on the right keyboard and returns to the idle loop. Thereafter, the operator can proceed the same as with the right overlay, actuating a succession of key programs as guided by the left overlay lights. The operator can shift freely back and forth between overlays depressing any key that is lighted. If he depresses an unlit key, the interrupt message is rejected by the Executive and error status light 3 is turned on to signal an error. (The blinking procedure external error light is also turned on.) The error condition is cleared by pressing a lighted (legal) key.

If the operator now desires to change the overlay on

one of the keyboards, say the left, he simply picks up the current left overlay and puts down the new one. Light L1 goes on and all other left key lights go off. However, the key programs and sub-routines associated with the new left overlay have not been loaded yet. Hence, if the operator changes his mind, he is free to pick up the new left overlay and put down the old one. In either case, he can then proceed to use the left overlay, as before, by pressing L1. The programs for the new (or old) left overlay will now be loaded, an initial set of left key lights will go on and the operator proceeds as before. Similar remarks apply to changing a right overlay.

When a new set of left overlay programs is to be loaded, the Executive first releases the memory space occupied by the old left overlay programs (if any) and then loads the new left overlay into this space. When a new set of right overlay programs is to be loaded, the Executive releases the memory space occupied by both the left and right overlays (if any). It reloads both the new right overlay programs and the current left overlay programs. This follows because space is so allocated that if both left and right overlay programs are in core, the left overlay programs will be allocated the space adjacent to the unallocated space. Thus if the new left overlay programs occupy more space than the old, there is room for expansion; if a new set of right overlay programs occupy more space than the old, both left and right overlay programs must be reallocated to make room in memory.

## C.2 DOCUS Library Organization

In normal CO-OP operation, all system programs and "permanent" job programs are stored on one magnetic tape, the "MCS tape". MCS itself is on this tape; so are all SCS programs

like CO-OP and job programs like FORTRAN, CODAP, the SORT program, etc. All subroutines employed by these programs are also on this one tape in addition to CO-OP utility programs, e.g., the core dump. In short, the MCS tape contains the entire CO-OP system and library.

The DOCUS system is contained on two magnetic tapes, the DOCUS MCS tape and the DOCUS library tape. The DOCUS tape is a CO-OP-type MCS tape with the DOCUS Executive assembled into MCS as explained above, a set of DOCUS subroutines in addition to the regular CO-OP library subroutines and certain other modifications to be discussed below.

The DOCUS library tape contains key programs and displays. The key programs may (and often do) call on subroutines stored on the MCS tape.

The following fundamental differences exist between the two tapes:

- A) A key program is loaded from the library tape into memory only if it is called or belongs to an overlay that is called. On the other hand, a subroutine is loaded if it is called or if it is referenced in any other subroutine that is to be loaded. Thus, loading subroutines from the MCS tape is a more complex process than loading key programs from the library tape; hence a more elaborate directory organization is required for efficient (1-pass) loading.
- B) Key programs may be added or deleted from the library tape by the DOCUS operator at his on-line

console. This updating procedure involves only one tape, i.e., the updated result is written on the original library tape rather than copied onto a second tape. To accomplish this, it is necessary to update the directory and then rewrite it on the tape. This, in turn, requires that the directory be at the end of the library tape; by contrast, the directory of the MCS tape is near the beginning of the tape.

- C) The library tape contains displays as well as key programs. These may be called by name. The MCS tape never contains displays in the DOCUS sense.

### C.3 Communication Areas

A number of words and areas are reserved for communication between the Executive and key programs in addition to communication between key programs. These reserved areas are in hard core, i.e., in the CO-OP Resident Monitor area.

---

The function of each of these areas contained in System Common are as follows:

- |          |   |
|----------|---|
| A) KNDEX | Contains the entry point addresses of the key programs of both the left and right overlays currently in memory. |
| B) LGWRD | Contains the character code and address of a symbol or line which has been light-gunned.                        |

G) DECODE	Determine condition of specified key or status light.
H) RDCUR	Read the coordinates of the current cursor position.
I) CLDSP	Clear the display memory within specified memory limits.
J) PACK	Pack 4 BR85 sequential codes into a 1604B word.
K) UNPACK	Opposite of PACK.
L) READ*A	The DOCUS equivalent of the CO-OP Monitor READ*. This is a generalized input routine.
M) WRITE*A	Same as READ*A except it applies to output.
N) LBOFF	Sets specified bank of lights to off condition.
O) SETLM	Sets the lights of a specified keyboard in accordance with a specified mask.
P) LGMODE	The value of this word indicates whether a light-gunned character may or may not be changed. LGMODE may be set by any key program before returning to the Executive to wait for a light-gun interrupt.
Q) WHATNEXT	The Executive sets this word before jumping to a key program in order to tell the key program



- whether this is an initial entry or an entry due to a light-gun or key interrupt.
- R) KEYSEQ      A counter for the number of times Executive enters a key program.
  - S) REFORMAT    A buffer which is primarily used to store a display page in unpacked form.
  - T) IOBUF        A buffer which is used to store a display page in packed form.
  - U) STBUF        Error messages are stored in this buffer in BCD format by key programs.
  - V) LIBIO        Contains the file and record number of the current position of the DOCUS library tape.
  - W) WORKIO       Same as LIBIO except applied to the work tape.

#### C.4      Key Program Calls on the Executive

There are a number of routines within the Executive on which a key program or subroutine may call. The functions of these routines are as follows:

- A) KRTRN        Point of return to Executive for all key programs. Checks status of a key program where the possible conditions are:

- 1) Wait for key or light interrupt;
  - 2) Key program requests Executive to load and execute a large user program and then return after restoring previous environment;
  - 3) Key program is complete, but its light settings should be maintained;
  - 4) Key program is complete; restore all initial light settings.
- 
- |            |  |
|------------|--|
| B) WDATA   | Write a block of display data from the 1604B to the BR85.                                  |
| C) RDATA   | Read a block of data from the BR85 into the 1604B.   |
| D) SETLT   | Set specified key or status light to on condition.   |
| E) RESETLT | Set specified key or status light to off condition.  |
| F) PROERR  | Turn on the "external procedure error" light in addition to a single (error) status light. |

## APPENDIX D

### References

- 1) Bauer, W. F., and Frank, W. L., DODDAC - An Integrated System for Data Processing, Interrogation and Display - Proceedings EJCC December 1961.
- 2) Culler-Fried - An On-Line Computing Center, RADC-TDR-63-160, July 1963, AD 414564, Contract AF30(602)-2762, Thompson Ramo Wooldrige, Inc.
- 3) The TRW Two Station, On-Line Scientific Computer, RADC-TDR-64-392, December 1964, AD 609720, Contract AF30(602)-3097, TRW/Space Technology Laboratories.
- 4) National Military Command System Support Center Display System. Volumes I, II, III, Contracts DA-49-146-XZ-211 and DCA-16.
- 5) Kameny, S. L., DeFlorio, G. P., and Varhaus, A. H. - General Purpose Display System, September 1964, SP-1699, AD 451859, System Development Corp.
- 6) Bennett, E., Haines, E., Summers, J., AESOP: A Prototype for On-Line User Control of Organizational Data Storage, Retrieval and Processing, Proceedings FJCC 1965.
- 7) Sutherland, I. E. - SKETCHPAD: A Man-Machine Graphical Communication System, January 1963, Report #296 and SJCC 1963, Lincoln Laboratory.

- 8) Allen, T. R. and Foote, J. E. - Input/Output Software Capability for a Man-Machine Communications and Image Processing System, FJCC 1964, General Motors Corporation.
- 9) Alpine, Graphical Design System, IBM Motion Picture.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)		
1. ORIGINATING ACTIVITY (Corporate author)		2a. REPORT SECURITY CLASSIFICATION
Informatics, Inc. 4720 Montgomery Lane Bethesda Md 20014		UNCLASSIFIED
		2b. GROUP
		N/A
3. REPORT TITLE		
DISPLAY ORIENTED COMPUTER USAGE SYSTEM		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)		
Interim Tech Report 1 Apr 65 - 1 Apr 66		
5. AUTHOR(S) (Last name, first name, initial)		
N/A		
6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
JUNE 1966	105	9
8a. CONTRACT OR GRANT NO. AF30(602)3500		8b. ORIGINATOR'S REPORT NUMBER(S)
a. PROJECT NO. 4594		None
c. Task No. 459402		8c. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.		RADC TR-66-324
10. AVAILABILITY/LIMITATION NOTICES		
This document is subject to special export controls and each transmittal to foreign government or foreign nationals may be made only with prior approval of RADC (EMLI), GAFB, N.Y. 13440.		
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY
		Rome Air Development Center (EMIRD) Griffiss Air Force Base, New York 13440.
13. ABSTRACT		
<p>The Display Oriented Computer Usage System (DOCUS) is a software system under which new on-line man/machine techniques for CRT console use are being designed and implemented for the non-programming user, on-line tools such as a File Maintenance and Query Language and a Text Manipulation Language are described. For the programming user a capability for on-line programming and debugging through the use of FORTRAN, CODAP, and a Display Oriented Language (DOL) is provided by the system. The basic framework in which these user tools operate is also provided by two groups of generalized function key programs. One group of keys, the General Operating Language gives the non-programming user a display manipulation, library control, and system communication facility. The second group of keys, the Procedure Implementation Language, allows the programmer to create displays and key programs and to build a repertoire of functions in a system and various user libraries.</p>		

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

16. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
PROGRAMMING DISPLAY CONSOLES MAN-MACHINE COMPUTERS						

## INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through \_\_\_\_\_."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through \_\_\_\_\_."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through \_\_\_\_\_."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification